Conformal Prediction in Hierarchical Classification with Constrained Representation Complexity

Thomas Mortier[1,5] **Alireza Javanmardi**[2,3] **Yusuf Sale**[2,3]
Eyke Hüllermeier[2,3,4] **Willem Waegeman**[5]

[1]Department of Environment, Ghent University
[2]Institute of Informatics, LMU Munich
[3]MCML, Munich
[4]DFKI (DSA), Kaiserslautern
[5]Dept. of Data Analysis and Mathematical Modelling, Ghent University

Abstract

Conformal prediction has emerged as a widely used framework for constructing valid prediction sets in classification and regression tasks. In this work, we extend the split conformal prediction framework to hierarchical classification, where prediction sets are commonly restricted to internal nodes of a predefined hierarchy, and propose two computationally efficient inference algorithms. The first algorithm returns internal nodes as prediction sets, while the second one relaxes this restriction. Using the notion of representation complexity, the latter yields smaller set sizes at the cost of a more general and combinatorial inference problem. Empirical evaluations on several benchmark datasets demonstrate the effectiveness of the proposed algorithms in achieving nominal coverage.

1 INTRODUCTION

In multi-class classification, a classifier can be uncertain about the predicted class label for a given test instance. In such cases, it can be beneficial to return set-valued predictions, i.e. sets of classes rather than individual classes. This is particularly relevant in hierarchical classification, where the class space is organised in a hierarchical structure, such as in medical diagnosis, where diseases are organised in a tree structure based on the International Classification of Diseases (ICD) (World Health Organization, 1978).

In hierarchical classification, set-valued predictions are often restricted to internal nodes of the hierarchy. Such predictions have a clear semantic interpretation and can be computed using efficient inference algorithms (Alex Freitas, 2007; Bi and Kwok, 2015; Rangwala and Naik, 2017; Yang et al., 2017; Wang et al., 2021; Valmadre, 2022). However, a restriction of this kind can affect efficiency when the classifier is uncertain between classes in different branches of the hierarchy. In these cases, predicting a single internal node typically yields large and uninformative sets.

To address this, some approaches allow any subset of classes as predictions, which improves flexibility but comes at the cost of higher semantic complexity and reduced interpretability (Oh, 2017). More recently, a set-based utility maximisation framework has been proposed that makes a compromise between the two aforementioned extremes by introducing the notion of representation complexity (Mortier et al., 2022). The assumption behind this notion is that inner nodes of a hierarchy have semantic meaning and can therefore be used as meaningful stand-alone predictions. By limiting the number of internal nodes used to represent a prediction, the proposed framework allows users to control the trade-off between efficiency and interpretability.

To illustrate the usefulness of representation complexity, consider the PlantCLEF 2015 dataset (Göeau et al., 2015). This dataset consists of over one hundred thousand images representing 1,000 species of trees, herbs, and ferns native to

Proceedings of the 29th International Conference on Artificial Intelligence and Statistics (AISTATS) 2026, Tangier, Morocco. PMLR: Volume 300.

Figure 1: A sample image of the *Lotus corniculatus* species from the PlantCLEF 2015 dataset (Göeau et al., 2015).

the Western European region and is characterised by significant class ambiguity, making accurate predictions on species level often impossible. In Figure 1, an example image is shown corresponding to the *Lotus corniculatus* species. Using our proposed algorithms, two set-valued predictions were computed, with a restricted representation complexity of $r = 1$ and $r \leq 3$, respectively. The former yields the root node of the hierarchy, i.e. all 1,000 species, and is completely uninformative. Increasing the representation complexity to three improves efficiency and yields the prediction $\{$*Lotus corniculatus*, *Tulipa sylvestris*, *Ficaria verna*$\}$, i.e., three visually-related species including the true label.

Contributions. In this work, we build further upon the concept of representation complexity and extend the split conformal prediction framework to the hierarchical multi-class classification setting. Conformal prediction provides valid and efficient set-valued predictions for any learning algorithm, without making any assumptions on the underlying data distribution (Vovk et al., 2005). In particular, given a trained (hierarchical) classifier, a desired coverage level $1-\alpha \in (0,1)$, calibration samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^{N} \subset \mathcal{X} \times \mathcal{Y}$ and a test sample $(\boldsymbol{x}_{N+1}, y_{N+1})$, drawn i.i.d. from an unknown distribution P, we would like to construct a prediction set $\hat{Y} \in 2^{\mathcal{Y}}$ with representation complexity r for the test instance $(\boldsymbol{x}_{N+1}, y_{N+1})$, such that the following marginal validity guarantee holds:

$$\mathbb{P}\left[y_{N+1} \in \hat{Y}(\boldsymbol{x}_{N+1})\right] \geq 1-\alpha\,, \text{ s.t. } R_{\mathcal{T}}(\hat{Y}) \leq r\,, \quad (1)$$

where $R_{\mathcal{T}}(\hat{Y})$ denotes the representation complexity of the set $\hat{Y}$, given some tree structure $\mathcal{T}$. The probability is taken over all $N+1$ samples and we require that (1) holds for any fixed α, N, r and P. Figure 2, which will be explained in detail later on, intuitively illustrates the concept of representation complexity. In this example of a hierarchical tree structure with eight classes, the set $\hat{Y} = \{1,2,4,7,8\}$ has a representation complexity of three, as it requires three nodes to represent this set: v_4, v_7 and v_{11}.

After reviewing some essentials on hierarchical probabilistic classification and conformal prediction in Section 2, we propose in Section 3 two algorithms that construct valid prediction sets, in the sense of satisfying (1). The first algorithm is designed for the restrictive case $r = 1$, while the second extends to $r > 1$. Moreover, we show that our algorithms possess distribution-free finite sample guarantees. Finally, in Section 4, we evaluate the proposed algorithms in terms of coverage and efficiency on a range of benchmark datasets.

2 BACKGROUND, RELATED WORK AND FORMAL DESCRIPTION OF EXISTING CONCEPTS

2.1 Hierarchical multi-class classification

In a standard multi-class classification setting, one assumes that training and test data are i.i.d. according to an unknown distribution $P(\boldsymbol{x}, y)$ on $\mathcal{X} \times \mathcal{Y}$, with $\mathcal{X}$ some instance space (e.g. images, documents, etc.) and $\mathcal{Y} = \{c_1, \ldots, c_K\}$ a class space consisting of K classes. Probabilistic multi-class classifiers estimate the conditional class probabilities $P(\cdot \,|\, \boldsymbol{x})$ over $\mathcal{Y}$, such that $0 \leq P(c \,|\, \boldsymbol{x}) \leq 1$ for all $c \in \mathcal{Y}$ and $\sum_{c \in \mathcal{Y}} P(c \,|\, \boldsymbol{x}) = 1$. This distribution can be estimated using a wide range of well-known probabilistic models, such as logistic regression, linear discriminant analysis, gradient boosting trees or neural networks with a softmax output layer. At prediction time, set-valued prediction algorithms, such as split conformal prediction, return sets $\hat{Y}$ that are subsets of $\mathcal{Y}$. The probability mass of such a set can be computed as $P(\hat{Y} \,|\, \boldsymbol{x}) = \sum_{c \in \hat{Y}} P(c \,|\, \boldsymbol{x})$.

In this paper, we will consider a hierarchical multi-class classification setting. Hence, we assume that a domain expert has defined a hierarchy over the class space, in the form of a tree structure $\mathcal{T}$ that in general contains M nodes. $\mathcal{V}_{\mathcal{T}} = \{v_1, \ldots, v_M\}$ will denote the set of nodes and every node identifies a set of classes. As special cases, the root v_1 represents the class space $\mathcal{Y}$, and the leaves represent individual classes; see Figure 2 for a simple example. In hierarchical classification, the strong restriction $\hat{Y} \in \mathcal{V}_{\mathcal{T}}$ is typically made for predicted sets; see e.g. Bi and Kwok (2015). The probability mass of such a set can be computed as $P(v \,|\, \boldsymbol{x}) = \sum_{c \in v} P(c \,|\, \boldsymbol{x})$, or by using the chain rule of probability:

$$P(v \,|\, \boldsymbol{x}) = \prod_{v' \in \text{Path}(v)} P(v' \,|\, \text{pa}(v'), \boldsymbol{x})\,, \quad (2)$$

where $\text{Path}(v)$ is a set of nodes on the path connecting the node v and the root of the tree

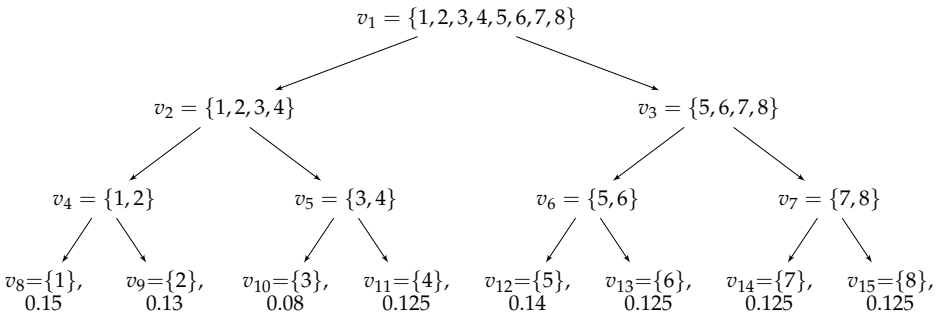

Figure 2: An example of a tree structure $\mathcal{T}$ with class space $\mathcal{Y} = \{1, \ldots, 8\}$ and nodes $\mathcal{V}_\mathcal{T} = \{v_1, \ldots, v_{15}\}$. The root v_1 represents the class space $\mathcal{Y}$ and leaves $\{v_8, \ldots, v_{15}\}$ represent the individual classes. The numbers in the leaf nodes represent the class probabilities for an instance $\boldsymbol{x}$.

structure. $\text{pa}(v)$ gives the parent of node v and $P(v \,|\, \text{pa}(v), \boldsymbol{x})$ represents the branch probability of node v given its parent $\text{pa}(v)$. Note that for the root node v_1 one has $P(v_1 \,|\, \text{pa}(v_1), \boldsymbol{x}) = 1$. In Figure 2, the branch probabilities of the root node v_1 are given by $P(v_2 \,|\, v_1, \boldsymbol{x}) = 0.485$ and $P(v_3 \,|\, v_1, \boldsymbol{x}) = 0.515$. In order to estimate the branch probabilities, one can train any multi-class probabilistic classifier in each internal node of the tree. Classical models of that kind include nested dichotomies (Fox, 1997; Frank and Kramer, 2004; Melnikov and Hüllermeier, 2018), conditional probability estimation trees (Beygelzimer et al., 2009) and probabilistic classifier trees (Dembczyński et al., 2016). In neural networks with a hierarchical softmax output layer, all nodes are trained simultaneously (Morin and Bengio, 2005).

2.2 Inference in hierarchical classification

In this work, we do not focus on the training algorithms. Instead, we assume that a probabilistic model $\hat{P}$ has been estimated, either with classical models or using a hierarchical factorization as in (2), and we focus on the prediction task. In particular, we would like to predict sets $\hat{Y} \in 2^\mathcal{Y}$ that satisfy (1), with a restriction on the representation complexity of the predicted set (Mortier et al., 2022). The representation complexity is defined as the minimal number of nodes needed to represent the set $\hat{Y}$ in the tree structure $\mathcal{T}$. More formally, let $\mathcal{S}_\mathcal{T}(\hat{Y})$ denote the set of all disjoint combinations of tree nodes that represent the set $\hat{Y}$:

$$\mathcal{S}_\mathcal{T}(\hat{Y}) = \left\{ \hat{V} \subset \mathcal{V}_\mathcal{T} : \bigcup_{v_i \in \hat{V}} v_i = \hat{Y} \wedge \bigcap_{v_i \in \hat{V}} v_i = \emptyset \right\}.$$

Then, we define the representation complexity of the prediction $\hat{Y}$ as

$$R_\mathcal{T}(\hat{Y}) = \min_{\hat{V} \in \mathcal{S}_\mathcal{T}(\hat{Y})} |\hat{V}|, \tag{3}$$

with $|\hat{V}|$ the cardinality of $\hat{V}$. For example, the set $\hat{Y} = \{1,2,4,7,8\}$ of the hierarchy shown in Figure 2 has representation complexity three, because three nodes are needed to represent this set: v_4, v_7 and v_{11}.

2.3 Randomized nested prediction sets with split conformal prediction

Now we will review a general procedure for valid set-valued predictions in flat classification (i.e. ignoring the hierarchy), following the work of Gupta et al. (2022); Romano et al. (2020); Angelopoulos et al. (2020). Compared to traditional conformal prediction—which starts from the notion of a nonconformity score—this procedure departs from a sequence of nested set-valued predictions, where the set size depends on a threshold τ. Furthermore, an independent calibration set is used to tune the threshold τ such that $(1-\alpha)$-coverage is guaranteed on future test samples. The use of a single calibration set is better known as split conformal prediction and gives rise to computationally efficient valid set-valued predictions (Papadopoulos et al., 2002; Lei et al., 2018).

More formally, let $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^N$ be an i.i.d. sequence of N samples from the unknown distribution P, and assume that these samples were not used for model training. Let $\hat{Y}(\boldsymbol{x}, u, \tau) : \mathcal{X} \times [0,1] \times \mathbb{R} \to 2^\mathcal{Y}$ be a set-valued predictor. In the spirit of Romano et al. (2020); Angelopoulos et al. (2020), the second argument u represents a random draw from a uniform distribution $\mathcal{U}(0,1)$ and is included to allow for randomized prediction sets. The third argument τ is a threshold that controls the size of the predicted set. Assume that the sets are indexed by τ and are nested:

$$\hat{Y}(\boldsymbol{x}, u, \tau_1) \subseteq \hat{Y}(\boldsymbol{x}, u, \tau_2) \qquad \text{if} \quad \tau_1 \leq \tau_2 \,. \tag{4}$$

In order to find the optimal threshold τ^* that guarantees $(1-\alpha)$-coverage, one needs to find the smallest τ such that the predicted set $\hat{Y}(\boldsymbol{x}, u, \tau)$

contains at least $\lceil (N+1)(1-\alpha) \rceil$ samples:

$$\tau^* = \inf \left\{ \tau \in [0,1] : |\{i : y_i \in \hat{Y}(\boldsymbol{x}_i, u_i, \tau)\}| \geq \lceil (1-\alpha)(N+1) \rceil \right\}. \quad (5)$$

The set-valued predictor $\hat{Y}(\boldsymbol{x}_{N+1}, u_{N+1}, \tau^*)$, with τ^* in (5), has marginal validity guarantees, as shown by the theorem below.

Theorem 2.1 (Marginal validity of nested conformal prediction (Angelopoulos et al., 2020))**.** *Assume an exchangeable sequence* $\{(\boldsymbol{x}_i, y_i, u_i)\}_{i=1}^{N+1}$ *and let* $\hat{Y}(\boldsymbol{x}, u, \tau)$ *be a set-valued predictor that satisfies (4). Furthermore, assume that* $\exists \tau \in \mathbb{R} : \hat{Y}(\boldsymbol{x}, u, \tau) = \mathcal{Y}$. *Then, for* τ^* *in (5) and any* $\alpha \in (0,1)$*, the following marginal coverage guarantee holds:*

$$1 - \alpha \leq \mathbb{P}\left[y_{N+1} \in \hat{Y}(\boldsymbol{x}_{n+1}, u_{N+1}, \tau^*) \right].$$

In the literature, several conformal prediction methods can be found that satisfy (4). For example, Romano et al. (2020) propose the following set-valued predictor, called adaptive prediction sets (APS):

$$\hat{Y}_{APS}(\boldsymbol{x}, u, \tau) = \left\{ y \in \mathcal{Y} : \hat{\rho}(y; \boldsymbol{x}) + u \cdot \hat{P}(y \mid \boldsymbol{x}) \leq \tau \right\}, \quad (6)$$

with $\hat{\rho}(y; \boldsymbol{x}) = \sum_{y' \in \mathcal{Y}} \hat{P}(y' \mid \boldsymbol{x}) \mathbb{1}_{\hat{P}(y' \mid \boldsymbol{x}) > \hat{P}(y \mid \boldsymbol{x})}$ the probability mass of the labels more likely than y. At the heart of their method is the randomization term $u \cdot \hat{P}(y \mid \boldsymbol{x})$, which is used to achieve exact nominal coverage. In addition, by means of the cumulative distribution $\hat{\rho}(y; \boldsymbol{x})$, this method allows to adapt effectively to complex data distributions, achieving a better conditional coverage compared to alternative approaches that rely on the mode of the distribution, such as the least ambiguous classifier proposed by Sadinle et al. (2019). Angelopoulos et al. (2020) propose a variant of the above method, called the regularized adaptive prediction sets (RAPS) method. By introducing a regularization term in (6), small set sizes are encouraged, especially when facing noisy probability estimates for classes with low probability.

Alternative conformal prediction methods have been introduced, focusing on minimizing set size (Sadinle et al., 2019; Gao et al., 2025) and achieving approximate coverage across the entire feature space (Foygel Barber et al., 2021; Cauchois et al., 2021; Romano et al., 2019). Goren et al. (2024) and Angelopoulos et al. (2023) propose conformal prediction methods that control the risk with respect to bounded non-increasing loss functions. Interestingly, these methods are also applicable to the setting of hierarchical classification, e.g. through the use of the tree-distance loss, introduced by Bi and Kwok (2015) as a way of evaluating set-valued predictions in multi-label classification. However, set-valued predictions of that kind lack meaningful and practical interpretation, compared to methods that rely on the traditional notion of coverage. Furthermore, these methods are not directly applicable to our work, as they do not allow the incorporation of representation complexity, given the properties of the loss functions above.

Gao et al. (2025) recently proposed a novel conformal predictor for regression, by implementing a dynamic programming algorithm that finds a union of k-intervals on the real line. When the output distribution is multi-modal, a union of k-intervals can be more informative and efficient compared to a single prediction interval (as obtained with most regression methods). Our work has a similar motivation, but adopted to hierarchical classification: when the classifier is uncertain between classes in different branches of the hierarchy, restricting the representation complexity to one is overly restrictive, while a higher complexity yields more efficient predictions.

3 PROPOSED METHODS

Building upon the concepts in Section 2.3, we are now ready to discuss two algorithms that provide valid set-valued predictions for hierarchical classification that satisfy (1) above. The first algorithm provides marginal validity guarantees in classical hierarchical classification settings, i.e. where sets are restricted to nodes of the tree structure, thus $R_{\mathcal{T}}(\hat{Y}) = 1$. The second algorithm provides marginal validity guarantees in restricted set-valued prediction settings, i.e. where the representation complexity of the predicted set is restricted to $R_{\mathcal{T}}(\hat{Y}) \leq r$ for a user-defined value r. Note that traditional conformal prediction in flat classification is obtained for the second algorithm when there is no restriction on the representation complexity (e.g. $r = K$).

3.1 Conformal restricted set-valued prediction

The first approach, called conformal restricted set-valued prediction (CRSVP), predicts sets that are restricted to nodes of the hierarchy, thus having representation complexity $R_{\mathcal{T}}(\hat{Y}) = 1$. Assume one has fitted a classifier using a training dataset as described in Section 2.1, and let $\hat{y}(\boldsymbol{x})$ denote the mode of the distribution $\hat{P}(. \mid \boldsymbol{x})$, i.e. the leaf node with highest probability mass. For our first approach, we consider the following set-valued predictor:

Algorithm 1 CRSVP calibration – **Input:** $\{(\boldsymbol{x}_i, y_i, u_i)\}_{i=1}^{N}, \hat{P}, \mathcal{V}_{\mathcal{T}}$, **Output:** Threshold in (5).

for $i = 1, \ldots, N$ **do**
  $\hat{Y} \leftarrow \arg\max_{c \in \mathcal{Y}} \hat{P}(c \mid \boldsymbol{x}_i)$
  $\hat{p}_{\hat{Y}} \leftarrow \hat{P}(\hat{Y} \mid \boldsymbol{x}_i), \hat{p}_{\hat{Y}'} \leftarrow 0$
  while $y_i \notin \hat{Y}$ **do**
    $\hat{p}_{\hat{Y}'} \leftarrow \hat{p}_{\hat{Y}}, \hat{p}_{\hat{Y}} \leftarrow \hat{P}(\text{pa}(\hat{Y}) \mid \boldsymbol{x}_i)$
    $\hat{Y}' \leftarrow \hat{Y}, \hat{Y} \leftarrow \text{pa}(\hat{Y})$
  end while
  $\tau_i \leftarrow \hat{p}_{\hat{Y}} - u_i \cdot (\hat{p}_{\hat{Y}} - \hat{p}_{\hat{Y}'})$
end for
$\tau^* \leftarrow$ the $\lceil (1-\alpha)(N+1) \rceil$-th largest value in $\{\tau_i\}_{i=1}^{N}$
return τ^*

Algorithm 2 CRSVP inference – **Input:** $\boldsymbol{x}, \tau^*, u, \hat{P}, \mathcal{V}_{\mathcal{T}}$, **Output:** Set-valued prediction in (7).

$\hat{Y} \leftarrow \arg\max_{c \in \mathcal{Y}} \hat{P}(c \mid \boldsymbol{x}), \hat{Y}' \leftarrow \emptyset$
$\hat{p}_{\hat{Y}} \leftarrow \hat{P}(\hat{Y} \mid \boldsymbol{x}), \hat{p}_{\hat{Y}'} \leftarrow 0$
while $\text{pa}(\hat{Y}') \neq \emptyset$ **do**
  if $\hat{p}_{\hat{Y}} \geq \tau^*$ **then**
    break
  end if
  $\hat{p}_{\hat{Y}'} \leftarrow \hat{p}_{\hat{Y}}, \hat{p}_{\hat{Y}} \leftarrow \hat{P}(\text{pa}(\hat{Y}) \mid \boldsymbol{x})$
  $\hat{Y}' \leftarrow \hat{Y}, \hat{Y} \leftarrow \text{pa}(\hat{Y})$
end while
if $\hat{p}_{\hat{Y}} - u \cdot (\hat{p}_{\hat{Y}} - \hat{p}_{\hat{Y}'}) \geq \tau^*$ **then**
  return $\hat{Y}'$
else
  return $\hat{Y}$
end if

$$\hat{Y}_1(\boldsymbol{x}, u, \tau) = \underset{\hat{Y} \in \text{Path}(\hat{y}(\boldsymbol{x}))}{\arg\max} \left\{ |\hat{Y}| : \hat{P}(\hat{Y} \mid \boldsymbol{x}) + u \cdot \hat{P}(\text{pa}(\hat{Y}) \setminus \hat{Y} \mid \boldsymbol{x}) \leq \tau \right\}, \quad (7)$$

with $\text{Path}(v)$ the path to the root, and $\text{pa}(v)$ the parent of a node v, as defined in Section 2. $\hat{P}(\hat{Y} \mid \boldsymbol{x})$ can be computed using (2), but also observe that the probability mass of an internal node v corresponds to $\hat{P}(v \mid \boldsymbol{x}) = \sum_{c \in v} \hat{P}(c \mid \boldsymbol{x})$, i.e. the sum of the probability masses of the leaf nodes that are descendants of v. However, computing the probability mass of an internal node and its associated set of classes in this way is computationally less efficient than using the chain rule in (2), when hierarchical classifiers have been fitted during training.

It is clear that the set-valued predictor in (7) satisfies the nestedness property in (4). Indeed, starting from the mode of the distribution $\hat{y}(\boldsymbol{x})$, every node on the path connecting that node and the root is nested by definition of the hierarchical tree structure $\mathcal{T}$. In line with Angelopoulos et al. (2020), the first term increases as we move up the tree, while the second term contains a random draw from the uniform distribution $\mathcal{U}(0, 1)$, for handling discrete jumps in probability mass when following the path towards the root. The randomization is needed to prevent over-coverage.

The algorithm for calculating the threshold in (5) using the calibration dataset is presented in Algorithm 1. This algorithm first computes the mode of the estimated class probabilities for each instance in the calibration dataset. For small hierarchies, this can be done using exhaustive search, but more efficient inference algorithms have been developed when the computational cost becomes a burden (Dembczyński et al., 2012; Kumar et al., 2013; Mena Waldo et al., 2015). Subsequently, starting from the mode, Algorithm 1 computes for every calibration instance the internal node, on the path to the root, that also includes the true class label. Nonconformity scores that incorporate a randomization component are constructed for these internal nodes, and the final threshold is deduced from the scores. Algorithm 1 has a worst case $\mathcal{O}(N K)$ time complexity when exhaustive search is applied in the first step.

The pseudocode for computing the set-valued prediction for a new test instance, as defined in (7), is shown in Algorithm 2. Given a hierarchical classifier, the computational complexity during test time is given by $\mathcal{O}(\log K)$.

3.2 Conformal set-valued prediction with representation complexity

It should be obvious that sets of representation complexity one quickly become very big, since they correspond to internal nodes of the hierarchy that exhibit a predefined coverage level. Therefore, we present a second approach, called conformal set-valued prediction with representation complexity (CRSVP-*r*), that relaxes the representation complexity constraint to $R_{\mathcal{T}}(\hat{Y}) \leq r$ with r a user-defined parameter. Let $S_k \equiv \{y^{(1)}, \ldots, y^{(k)}\}$ denote the top-k classes for $\boldsymbol{x}$ when sorting the classes according to $\hat{P}(y \mid \boldsymbol{x})$, similar as in (6). We define the following sequence of optimization problems for any $k \in \{1, ..., K\}$:

$$A_r(S_k; \boldsymbol{x}) = \underset{\substack{\hat{Y} \in 2^{\mathcal{Y}} : R_{\mathcal{T}}(\hat{Y}) \leq r, \\ A_r(S_{k-1}; \boldsymbol{x}) \cup y^{(k)} \subseteq \hat{Y}}}{\arg\min} |\hat{Y}| - \hat{P}(\hat{Y} \mid \boldsymbol{x}), \quad (8)$$

with $A_r(S_1; \boldsymbol{x}) = y^{(1)}$. Remark that these optimization problems have to be solved in the correct order, because of the nestedness condition. For fixed k we refer to this optimization problem as *finding the set of common ancestors*. It can be thought of as a variant of the well-known lowest

Algorithm 3 CRSVP-r calibration – **Input:** $\{(\boldsymbol{x}_i, y_i, u_i)\}_{i=1}^N, r, \hat{P}, \mathcal{V}_\mathcal{T}$, **Output:** Threshold in (5).

for $i = 1, \ldots, N$ **do**
  $k \leftarrow 1$
  $\hat{Y}^{(k-1)} \leftarrow \emptyset, \hat{Y}^{(k)} \leftarrow \arg\max_{c \in \mathcal{Y}} \hat{P}(c \mid \boldsymbol{x}_i)$
  $\hat{p}_{\hat{Y}^{(k-1)}} \leftarrow 0, \hat{p}_{\hat{Y}^{(k)}} \leftarrow \hat{P}(\hat{Y}^{(k)} \mid \boldsymbol{x}_i)$
  while $k < K$ **do**
    if $y_i \notin \hat{Y}^{(k)}$ **then**
      $\hat{Y}^{(k)'} \leftarrow A_r(S_k; \boldsymbol{x}_i)$ (by means of Algorithm 5)
      $\hat{p}_{\hat{Y}^{(k)'}} \leftarrow \hat{P}(\hat{Y}^{(k)'} \mid \boldsymbol{x}_i)$
      if $|\hat{Y}^{(k)'}| \neq |\hat{Y}^{(k-1)}|$ **then**
        $\hat{Y}^{(k)}, \hat{p}_{\hat{Y}^{(k)}} \leftarrow \hat{Y}^{(k)'}, \hat{p}_{\hat{Y}^{(k)'}}$
      end if
      $k \leftarrow k + 1$
    else
      break
    end if
  end while
  $\tau_i \leftarrow \hat{p}_{\hat{Y}^{(k)}} - u_i \cdot (\hat{p}_{\hat{Y}^{(k)}} - \hat{p}_{\hat{Y}^{(k-1)}})$
end for
$\tau^* \leftarrow$ the $\lceil (1-\alpha)(N+1) \rceil$-th largest value in $\{\tau_i\}_{i=1}^N$
return τ^*

Algorithm 4 CRSVP-r inference – **Input:** $\boldsymbol{x}, \tau^*, u, r, \hat{P}, \mathcal{V}_\mathcal{T}$, **Output:** Set-valued prediction in (9).

$k \leftarrow 1$
$\hat{Y}^{(k-1)} \leftarrow \emptyset, \hat{Y}^{(k)} \leftarrow \arg\max_{c \in \mathcal{Y}} \hat{P}(c \mid \boldsymbol{x})$
$\hat{p}_{\hat{Y}^{(k-1)}} \leftarrow 0, \hat{p}_{\hat{Y}^{(k)}} \leftarrow \hat{P}(\hat{Y}^{(k)} \mid \boldsymbol{x})$
while $k < K$ **do**
  if $y^{(k)} \notin \hat{Y}^{(k-1)}$ **then**
    $\hat{Y}^{(k)'} \leftarrow A_r(S_k; \boldsymbol{x})$ (by means of Algorithm 5)
    $\hat{p}_{\hat{Y}^{(k)'}} \leftarrow \hat{P}(\hat{Y}^{(k)'} \mid \boldsymbol{x})$
    if $|\hat{Y}^{(k)'}| \neq |\hat{Y}^{(k-1)}|$ **then**
      $\hat{Y}^{(k)}, \hat{p}_{\hat{Y}^{(k)}} \leftarrow \hat{Y}^{(k)'}, \hat{p}_{\hat{Y}^{(k)'}}$
    end if
    if $\hat{p}_{\hat{Y}^{(k)}} \geq \tau^*$ **then**
      break
    end if
    $k \leftarrow k + 1$
  end if
end while
if $\hat{p}_{\hat{Y}^{(k)}} - u \cdot (\hat{p}_{\hat{Y}^{(k)}} - \hat{p}_{\hat{Y}^{(k-1)}}) \geq \tau^*$ **then**
  return $\hat{Y}^{(k-1)}$
else
  return $\hat{Y}^{(k)}$
end if

Algorithm 5 Dynamic programming solution for set of lowest common ancestors – **Input:** $\boldsymbol{x}, S_k, r, \hat{P}, \mathcal{V}_\mathcal{T}, A_r(S_{k-1}; \boldsymbol{x})$, **Output:** $A_r(S_k; \boldsymbol{x})$

$\mathcal{Q} \leftarrow \emptyset$ ▷ initialize a queue for visiting nodes
for all $v \in \mathcal{V}_\mathcal{T}$ **do**
  $s^1(v) \leftarrow \emptyset, \ldots, s^r(v) \leftarrow \emptyset$
  if $|v| = 1 \wedge v \cap S_k \neq \emptyset$ **then**
    if $\mathrm{pa}(v) \notin \mathcal{Q}$ **then**
      $\mathcal{Q}$.add($\mathrm{pa}(v)$)
    end if
    $s^1(v) \leftarrow v, \ldots, s^r(v) \leftarrow v$ ▷ initialize leaves that overlap with S_k
  end if
end for
while $\mathcal{Q}$ is not empty **do**
  $v \leftarrow \mathcal{Q}$.pop()
  $T \leftarrow \{v' : v' \in \mathrm{ch}(v) \wedge v' \cap S_k \neq \emptyset\}$ ▷ visit all children that overlap with S_k
  for $r_i = 1$ to r **do**
    if $|T| > r_i$ **then**
      $s^{r_i}(v) \leftarrow v$
    else
      $M \leftarrow$ all compositions of r_i into $|T|$ elements
      $u^* \leftarrow +\infty$
      for $(i_0, \ldots, i_{|T|-1}) \in M$ **do**
        $s \leftarrow s^{i_0}(T[0]) \cup \cdots \cup s^{i_{|T|-1}}(T[|T|-1])$
        if $|s| - \hat{p}_s < u^*$ **then**
          if v is the root **then**
            if $A_r(S_{k-1}; \boldsymbol{x}) \subseteq s$ **then**
              $s^{r_i}(v) \leftarrow s$
              $u^* \leftarrow |s| - \hat{p}_s$
            end if
          else
            $s^{r_i}(v) \leftarrow s$
            $u^* \leftarrow |s| - \hat{p}_s$
          end if
        end if
      end for
    end if
  end for
  if $\mathrm{pa}(v) \notin \mathcal{Q} \wedge \mathrm{pa}(v) \neq \emptyset$ **then**
    $\mathcal{Q}$.add($\mathrm{pa}(v)$)
  end if
end while
return $s^r(v_1)$

common ancestor problem (Aho et al., 1973; Harel and Tarjan, 1984; Bender and Farach-Colton, 2000; Dash et al., 2013), i.e. instead of considering a single common ancestor of a set of leaves in a tree, we are looking for a set of r non-overlapping ancestors whose descendants form a set of representation complexity r. Moreover, we introduce the nested set condition, and the second term in the objective function is also not considered in the lowest common ancestor problem. Those two changes are needed to make sure that (1) the solutions for increasing k are nested, and (2) the solution for fixed k is unique. Since probabilities are bounded between zero and one, the second term only plays a role for sets that have the same cardinality (i.e. the two terms are lexicographically ordered). As an example, consider the hierarchy depicted in Figure 2. For $S_3 = \{1, 2, 5\}$, the lowest common ancestor is given by $A_1(S_3; \boldsymbol{x}) = \{1, 2, 3, 4, 5, 6, 7, 8\}$. The lowest set of common ancestors with representation complexity two is $A_2(S_3; \boldsymbol{x}) = \{1, 2, 5\}$, illustrating that set

size can be drastically reduced by increasing the representation complexity.

Let $\hat{Y}^{(1)}, \ldots, \hat{Y}^{(K)}$ denote the ordered sequence of unique lowest common ancestors $A_r(S_1; \boldsymbol{x}), \ldots, A_r(S_K; \boldsymbol{x})$. We use this sequence to introduce the following set-valued predictor:

$$\hat{Y}_{\leq r}(\boldsymbol{x}, u, \tau) = \underset{\hat{Y}^{(k)} \in \{\hat{Y}^{(1)}, \ldots, \hat{Y}^{(K)}\}}{\arg\max} \left\{ |\hat{Y}^{(k)}| : \hat{P}(\hat{Y}^{(k-1)} \mid \boldsymbol{x}) + u \cdot \hat{P}(\hat{Y}^{(k)} \setminus \hat{Y}^{(k-1)} \mid \boldsymbol{x}) \leq \tau \right\}. \quad (9)$$

Due to the nested set constraint in (8), the set-valued predictor in (9) satisfies the nestedness property in (4). Thus, this set-valued predictor automatically has the classical marginal coverage guarantees of conformal prediction. Similar as for the first approach, the randomization term $u \cdot \hat{P}(\hat{Y}^{(k)} \setminus \hat{Y}^{(k-1)} \mid \boldsymbol{x})$ is again included to prevent over-coverage.

Let us now discuss the algorithmic aspects of computing the set of lowest common ancestors in (8). At first impression, the combinatorial optimization problem in (8) looks challenging, but it can be solved in an efficient manner with a divide-and-conquer algorithm that is described in Algorithm 5.

For every internal node of the hierarchy, the optimization problem can be broken down into simpler subproblems of the same type. As an example, consider in a predefined hierarchy an internal node v' that has four children (v'_1, v'_2, v'_3, v'_4) containing at least one element of S_k, and assume that we aim to find the set of lowest common ancestors with representation complexity three for a set of classes S_k that are all descendants of v'. As potential divisions of the representation complexity, one could find one common ancestor as descendants of children v'_1, v'_3, and v'_4, or as descendants of children v'_1, v'_2, and v'_4, or as two descendants of v'_1 combined with one descendant of v'_4, etc. In fact, many combinations are possible. Speaking in formal combinatorial terminology, one has to consider for the node v all *compositions* of the integer three into four elements. For each of these compositions, the optimization problem can be recursively divided into smaller problems with lower representation complexity, where the node v is replaced by each of its non-zero children in the composition. However, implementing such a strategy in a recursive manner would heavily blow up the computations, since each of the smaller subproblems would need to be solved many times.

A dynamic programming implementation, which avoids recursion by solving the smaller subproblems in a bottom-up manner, before tackling the more challenging optimization problems higher up in the hierarchy, is able to solve (8) in an efficient manner for small values of r. The pseudocode of such an implementation is described in Algorithm 5. For every internal node, r local optimization problems are solved by varying the local representation complexity r_i from one to r, and the solutions of these optimization problems are stored. By visiting children before their parents get analysed, one can guarantee that all needed quantities are known when the different compositions in an internal node need to be investigated. The most critical step in Algorithm 5 is line 20, where all compositions of the current representation complexity r_i into $|T|$ elements are considered (with T the set of all children that contain at least one element of S_k). Strictly speaking, this step has a runtime that is exponential in r. However, in practical situations, one would only be interested in sets with a representation complexity lower than five, so Algorithm 5 in general computes the exact solution to (8) in an efficient manner.

The pseudocode for calculating the threshold in (5) and set-valued predictions in (9) is presented in Algorithm 3 and 4. The computational complexity during test time is dominated by the computation of the set of lowest common ancestors in (8), which needs to be computed for every k until the stopping criterion in Algorithm 4 is satisfied. Furthermore, the worst-case time complexity of calculating the set of lowest common ancestors in Algorithm 5 is upper bounded by $\mathcal{O}(K^2 r^d)$, with d the maximum out-degree of the tree $\mathcal{T}$. This is small for regimes that are practically useful (fixed $r \leq 3$ and moderate d).

Table 1: Summary statistics and top-1 performance for all datasets. Notation: K – number of classes, N – number of samples, Top-1 acc. – top-1 accuracy of classifier.

DATASET	K	N_{train}	N_{cal}	N_{test}	TOP-1 ACC.
CIFAR-10	10	50000	5000	5000	0.8817
AMB	93	12781	1918	1917	0.8503
CALTECH-101	97	4338	2169	2169	0.9039
DBPEDIA	219	276945	30397	30397	0.9388
CALTECH-256	256	14890	7445	7445	0.7578
PLANTCLEF 2015	1000	91758	10723	10723	0.4156

4 EXPERIMENTAL RESULTS

In this section, different set-valued predictors are compared for solving problem (1) across six benchmark datasets, focusing on coverage, efficiency, and representation complexity. Summary statistics for these datasets are presented in Table 1. For all datasets, a predefined hierarchy $\mathcal{T}$ is extracted from the provided hierarchical labels. For detailed

information about the experimental setup, we refer to Section A in the technical appendix.

In particular, we will compare various set-valued predictors that solve problem (1) across six datasets listed in Table 1: CIFAR-10 (Krizhevsky et al., 2010), Caltech-101 (Li et al., 2003), Caltech-256 (Griffin et al., 2007), PlantCLEF 2015 (Göeau et al., 2015), the Allen Mouse Brain (AMB) single-cell transcriptomics dataset (Tasic et al., 2018) and DBPedia (Ofer, 2019). The first four datasets are image datasets from the computer vision domain. The AMB dataset contains gene expression profiles from the mouse neocortex along with corresponding cell types and a hierarchical structure. The DBpedia text dataset contains extracted structured textual content from Wikipedia. Specifically, we evaluate the following set-valued predictors: CRSVP and CRSVP-3. Additionally, we demonstrate the usefulness of randomized prediction sets by considering the following naïve (N) set-valued predictors: NCRSVP and NCRSVP-3. These correspond to setting u to zero when calculating the threshold in (5) and constructing the prediction sets in (7) and (9), respectively. Finally, we include results for three baseline methods that produce valid set-valued predictions in flat classification (i.e. ignoring the hierarchy): the least ambiguous classifier (LAC), as proposed by Sadinle et al. (2019); adaptive prediction sets (APS), as defined in (6); and naïve prediction sets (NPS), which correspond to APS without randomization (i.e. setting u to zero). The results presented below are obtained using a flat classifier. We do not use hierarchical classifiers, as our dynamic programming approach relies on a bottom-up traversal of the tree, and incorporating hierarchical classifiers would increase computational complexity. Moreover, Mortier et al. (2022) report no significant improvement in predictive performance when using hierarchical classifiers compared to flat ones. Nevertheless, hierarchical classifiers can still be applied within our framework, as discussed in Section 2.1. During inference, we resample the calibration and test sets 10 times and use a confidence level of 90%.

The most important results are summarized in Table 2. For each experiment, we report (marginal) coverage, efficiency, and representation complexity. Coverage is defined as the proportion of samples for which the true class is included in the prediction set. Efficiency is defined as the average size of the set-valued prediction. The results clearly indicate that naïve set-valued predictors fail to deliver prediction sets with exact coverage, thereby highlighting the importance of randomized prediction sets. Moreover, increasing the representation complexity generally improves efficiency, demonstrating its practical value. In extreme cases, when representation complexity is unrestricted, such as with LAC, NPS, and APS, optimal performance in terms of efficiency is observed. However, this comes at the cost of significantly increased representation complexity in the prediction sets, in particular for large K, which may not be practical when predictions need to adhere to a predefined hierarchy.

Note that imprecise predictions for low representation complexity are typically due to uncertainty in the predicted probabilities and/or characteristics of the hierarchical structure, such as inconsistency and depth. Specifically, when uncertainty spans distinct branches of the hierarchy, predictions tend to correspond to higher-level nodes for lower representation complexity. This suggests that the hierarchy may not be well-suited for certain samples. Additionally, the depth of the hierarchical structure also plays a critical role in hierarchical classification tasks with a large number of classes. In such cases, a shallow hierarchy can lead to imprecise predictions for low representation complexity, as internal nodes in shallow trees often have many children. For example, the PlantCLEF 2015 dataset, characterized by 1,000 classes and a shallow hierarchy (i.e. taxonomy consisting of a family, genus and species level), requires a higher representation complexity in order to improve efficiency.

Finally, higher representation complexity becomes valuable for datasets with many classes and high uncertainty. In this case, traditional hierarchical classification tends to return imprecise predictions, i.e. nodes higher up in the hierarchy. By restricting representation complexity, predictions can span multiple nodes lower in the hierarchy, thereby improving efficiency. This is demonstrated by comparing CRSVP and CRSVP-3 on the PlantCLEF 2015 dataset in Table 1 and Figure 3. Furthermore, Figure 3 reveals a clear trade-off between representation complexity and efficiency for the PlantCLEF 2015 dataset. In Section B of the technical appendix, we present additional results obtained for (N)CRSVP-1 and (N)CRSVP-2, and provide additional insights regarding conditional coverage and the relationship between representation complexity and efficiency.

5 CONCLUSION

In this work, we extended the split conformal prediction framework to hierarchical classification

Table 2: Results for CIFAR-10, AMB, Caltech-101, DBpedia, Caltech-256 and PlantCLEF 2015. Coverage, efficiency, and representation complexity for the following unrestricted set-valued predictors: LAC, NPS, APS, and restricted set-valued predictors: NCRSVP, CRSVP, NCRSVP-3, and CRSVP-3. The confidence level is set to 90%, and calibration and test sets are resampled 10 times.

Dataset	Alg.	LAC	NPS	APS	NCRSVP	CRSVP	NCRSVP-3	CRSVP-3
CIFAR-10	Cov.	0.899 ± 0.005	0.997 ± 0.001	0.899 ± 0.003	1.000 ± 0.000	0.899 ± 0.005	0.997 ± 0.001	0.899 ± 0.003
	Size	1.473 ± 0.023	5.125 ± 0.058	1.849 ± 0.019	10.00 ± 0.000	3.899 ± 0.049	5.861 ± 0.064	1.946 ± 0.025
	R.C.	1.451 ± 0.021	3.552 ± 0.017	1.824 ± 0.015	1.000 ± 0.000	1.000 ± 0.000	2.368 ± 0.008	1.691 ± 0.009
AMB	Cov.	0.899 ± 0.010	1.000 ± 0.000	0.900 ± 0.009	1.000 ± 0.000	0.900 ± 0.009	1.000 ± 0.000	0.899 ± 0.009
	Size	1.128 ± 0.020	23.98 ± 1.195	1.685 ± 0.055	50.85 ± 1.304	4.856 ± 0.261	39.26 ± 1.574	2.184 ± 0.063
	R.C.	1.132 ± 0.019	17.38 ± 0.702	1.776 ± 0.051	1.000 ± 0.000	1.000 ± 0.000	1.829 ± 0.009	1.394 ± 0.018
Caltech-101	Cov.	0.900 ± 0.007	1.000 ± 0.000	0.900 ± 0.006	1.000 ± 0.000	0.901 ± 0.005	1.000 ± 0.000	0.901 ± 0.006
	Size	0.920 ± 0.008	96.54 ± 0.065	1.165 ± 0.015	96.25 ± 0.049	4.400 ± 0.254	63.61 ± 0.517	1.784 ± 0.086
	R.C.	1.000 ± 0.000	1.454 ± 0.072	1.251 ± 0.015	1.000 ± 0.000	1.000 ± 0.000	1.306 ± 0.008	1.133 ± 0.007
DBpedia	Cov.	0.899 ± 0.001	1.000 ± 0.000	0.901 ± 0.002	0.999 ± 0.000	0.901 ± 0.002	0.999 ± 0.000	0.901 ± 0.001
	Size	0.931 ± 0.002	59.17 ± 0.272	11.33 ± 0.287	162.9 ± 0.492	26.62 ± 0.417	127.9 ± 0.501	17.71 ± 0.423
	R.C.	1.000 ± 0.000	53.18 ± 0.257	11.90 ± 0.277	1.000 ± 0.000	1.000 ± 0.000	1.985 ± 0.004	1.380 ± 0.005
Caltech-256	Cov.	0.900 ± 0.004	0.999 ± 0.000	0.901 ± 0.004	0.999 ± 0.000	0.900 ± 0.003	1.000 ± 0.000	0.901 ± 0.003
	Size	1.931 ± 0.040	62.25 ± 1.330	3.640 ± 0.099	208.4 ± 1.780	44.69 ± 1.252	163.6 ± 2.070	20.30 ± 0.830
	R.C.	1.926 ± 0.040	50.12 ± 0.996	3.680 ± 0.098	1.000 ± 0.000	1.000 ± 0.000	1.632 ± 0.005	1.498 ± 0.009
PlantCLEF 2015	Cov.	0.899 ± 0.003	0.970 ± 0.001	0.899 ± 0.003	1.000 ± 0.000	0.900 ± 0.002	1.000 ± 0.000	0.899 ± 0.004
	Size	25.50 ± 0.450	123.6 ± 2.026	44.12 ± 0.994	998.9 ± 0.224	520.9 ± 4.745	995.3 ± 0.492	389.7 ± 5.898
	R.C.	24.33 ± 0.419	104.4 ± 1.560	40.19 ± 0.836	1.000 ± 0.000	1.000 ± 0.000	1.006 ± 0.001	1.632 ± 0.010

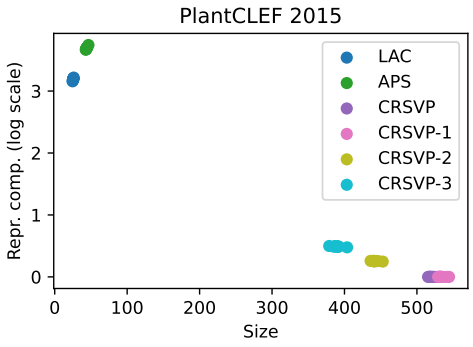

Figure 3: Trade-off between representation complexity (log scale) and efficiency for PlantCLEF 2015. The confidence level is set to 90%, and calibration and test sets are resampled 10 times.

by introducing two novel set-valued prediction algorithms. The first algorithm generates valid set-valued predictions restricted to single nodes within a predefined hierarchy. We argued that this restriction can be limiting in certain applications. To address this limitation, we introduced a second algorithm that relaxes this constraint by incorporating the notion of representation complexity. Empirical evaluations on multiple benchmark datasets demonstrate the effectiveness of the proposed algorithms in achieving exact nominal coverage.

The main purpose of bounding representation complexity is to ensure the interpretability and semantic meaningfulness of predictions. However, it may also have other benefits. In particular, we conjecture that it may serve the purpose of regularization and hence increase accuracy in cases where class probabilities are poorly estimated. In such cases, flat prediction sets tend to be scattered across the entire hierarchy, and this scattering is suppressed by bounding representation complexity. We plan to elaborate on this aspect more closely in future work. Another interesting direction for future research is to generalize our methods to more complex structures, such as directed acyclic graphs.

Acknowledgements

Yusuf Sale is supported by the DAAD program Konrad Zuse Schools of Excellence in Artificial Intelligence, sponsored by the Federal Ministry of Education and Research. Alireza Javanmardi gratefully acknowledges funding by the Klaus Tschira Stiftung (project 00.019.2024). Willem Waegeman received funding from the Flemish government under the Flanders AI research (FlAIR) program.

References

Aho, A. V., Hopcroft, J. E., and Ullman, J. D. (1973). On finding lowest common ancestors in trees. In *Proceedings of the fifth annual ACM symposium on Theory of computing*, pages 253–265.

Alex Freitas, A. d. C. (2007). A tutorial on hierarchical classification with applications in bioinformatics. In *Research and Trends in Data Mining Technologies and Applications*,, pages 175–208.

Angelopoulos, A., Bates, S., Fisch, A., Lei, L., and Schuster, T. (2023). Conformal risk control.

Angelopoulos, A. N., Bates, S., Malik, J., and Jordan, M. I. (2020). Uncertainty sets for image classifiers using conformal prediction. *ArXiv*, abs/2009.14193.

Bender, M. A. and Farach-Colton, M. (2000). The lca problem revisited. In *LATIN 2000: Theoretical Informatics: 4th Latin American Symposium, Punta del Este, Uruguay, April 10-14, 2000 Proceedings 4*, pages 88–94. Springer.

Beygelzimer, A., Langford, J., Lifshits, Y., Sorkin, G., and Strehl, A. (2009). Conditional probability tree estimation analysis and algorithms. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 51–58.

Bi, W. and Kwok, J. (2015). Bayes-optimal hierarchical multilabel classification. *IEEE Transactions on Knowledge and Data Engineering*, 27.

Cauchois, M., Gupta, S., and Duchi, J. C. (2021). Knowing what you know: valid and validated confidence sets in multiclass and multilabel prediction. *Journal of machine learning research*, 22(81):1–42.

Dash, S. K., Scholz, S.-B., Herhut, S., and Christianson, B. (2013). A scalable approach to computing representative lowest common ancestor in directed acyclic graphs. *Theoretical Computer Science*, 513:25–37.

Dembczyński, K., Kotłowski, W., Waegeman, W., Busa-Fekete, R., and Hüllermeier, E. (2016). Consistency of probabilistic classifier trees. In *ECML/PKDD*.

Dembczyński, K., Waegeman, W., and Hüllermeier, E. (2012). An analysis of chaining in multi-label classification. In *ECAI 2012*, pages 294–299. IOS Press.

Feldman, S., Bates, S., and Romano, Y. (2021). Improving conditional coverage via orthogonal quantile regression. *Advances in neural information processing systems*, 34:2060–2071.

Fox, J. (1997). *Applied regression analysis, linear models, and related methods*. Sage.

Foygel Barber, R., Candes, E. J., Ramdas, A., and Tibshirani, R. J. (2021). The limits of distribution-free conditional predictive inference. *Information and Inference: A Journal of the IMA*, 10(2):455–482.

Frank, E. and Kramer, S. (2004). Ensembles of nested dichotomies for multi-class problems. In *Proceedings of the Twenty-first International Conference on Machine Learning*, ICML '04.

Gao, C., Shan, L., Srinivas, V., and Vijayaraghavan, A. (2025). Volume optimality in conformal prediction with structured prediction sets. In *Forty-second International Conference on Machine Learning*.

Göeau, H., Joly, A., and Pierre, B. (2015). Lifeclef plant identification task 2015. *CLEF Working Notes*, 2015.

Goren, S., Galil, I., and El-Yaniv, R. (2024). Hierarchical selective classification. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Griffin, G., Holub, A., and Perona, P. (2007). Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology.

Gupta, C., Kuchibhotla, A. K., and Ramdas, A. (2022). Nested conformal prediction and quantile out-of-bag ensemble methods. *Pattern Recognition*, 127:108496.

Harel, D. and Tarjan, R. E. (1984). Fast algorithms for finding nearest common ancestors. *siam Journal on Computing*, 13(2):338–355.

Krizhevsky, A., Nair, V., and Hinton, G. E. (2010). Cifar-10 (canadian institute for advanced research). Technical report, Canadian Institute for Advanced Research.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2017). Imagenet classification with deep convolutional neural networks. *Communications of the ACM*, 60(6):84–90.

Kumar, A., Vembu, S., Menon, A. K., and Elkan, C. (2013). Beam search algorithms for multilabel learning. *Machine learning*, 92:65–89.

Lei, J., G'Sell, M., Rinaldo, A., Tibshirani, R. J., and Wasserman, L. (2018). Distribution-free predictive inference for regression. *Journal of the American Statistical Association*, 113(523):1094–1111.

Li, F.-F., Andreetto, M., and Ranzato, M. (2003). Caltech-101 image dataset. Technical report, California Institute of Technology.

Melnikov, V. and Hüllermeier, E. (2018). On the effectiveness of heuristics for learning nested dichotomies: an empirical analysis. *Machine Learning*, 107(8–10):1537–1560.

Mena Waldo, D., Montañés Roces, E., Quevedo Pérez, J. R., Coz Velasco, J. J. d., et al. (2015). Using a* for inference in probabilistic classifier chains. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence*

(IJCAI 2015). Association for the Advancement of Artificial Intelligence.

Morin, F. and Bengio, Y. (2005). Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics.

Mortier, T., Hüllermeier, E., Dembczyński, K., and Waegeman, W. (2022). Set-valued prediction in hierarchical classification with constrained representation complexity. In *Uncertainty in Artificial Intelligence*, pages 1392–1401. PMLR.

Mortier, T., Wydmuch, M., Dembczyński, K., and Waegeman, W. (2021). Efficient set-valued prediction in multi-class classification. *KDD*, 35:1435–1469.

Ofer, D. (2019). Dbpedia classes.

Oh, S. (2017). Top-k hierarchical classification. In *AAAI*, pages 2450–2456. AAAI Press.

Papadopoulos, H., Proedrou, K., Vovk, V., and Gammerman, A. (2002). Inductive confidence machines for regression. In *Machine learning: ECML 2002: 13th European conference on machine learning Helsinki, Finland, August 19–23, 2002 proceedings 13*, pages 345–356. Springer.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8026–8037.

Rangwala, H. and Naik, A. (2017). Large scale hierarchical classification: foundations, algorithms and applications. In *The European Conference on ML and Principles and Practice of Knowledge Discovery in Databases*.

Romano, Y., Patterson, E., and Candes, E. (2019). Conformalized quantile regression. *Advances in neural information processing systems*, 32.

Romano, Y., Sesia, M., and Candes, E. (2020). Classification with valid and adaptive coverage. *Advances in Neural Information Processing Systems*, 33:3581–3591.

Rossellini, R., Barber, R. F., and Willett, R. (2024). Integrating uncertainty awareness into conformalized quantile regression. In *International Conference on Artificial Intelligence and Statistics*, pages 1540–1548. PMLR.

Sadinle, M., Lei, J., and Wasserman, L. (2019). Least ambiguous set-valued classifiers with bounded error levels. *Journal of the American Statistical Association*, 114(525):223–234.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). Mobilenetv2: Inverted residuals and linear bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520.

Tasic, B., Yao, Z., Graybuck, L. T., Smith, K. A., Nguyen, T. N., Bertagnolli, D., Goldy, J., Garren, E., Economo, M. N., Viswanathan, S., et al. (2018). Shared and distinct transcriptomic cell types across neocortical areas. *Nature*, 563(7729):72–78.

Theunissen, L., Mortier, T., Saeys, Y., and Waegeman, W. (2024). Uncertainty-aware single-cell annotation with a hierarchical reject option. *Bioinformatics*, 40(3):btae128.

Valmadre, J. (2022). Hierarchical classification at multiple operating points. *Advances in Neural Information Processing Systems*, 35:18034–18045.

Vovk, V., Gammerman, A., and Shafer, G. (2005). *Algorithmic learning in a random world*, volume 29. Springer.

Wang, Y., Wang, Z., Hu, Q., Zhou, Y., and Su, H. (2021). Hierarchical semantic risk minimization for large-scale classification. *IEEE Transactions on Cybernetics*, 52(9):9546–9558.

World Health Organization, e. a. (1978). *International classification of diseases:[9th] ninth revision, basic tabulation list with alphabetic index*. World Health Organization.

Yang, G., Destercke, S., and Masson, M.-H. (2017). Cautious classification with nested dichotomies and imprecise probabilities. *Soft Computing*, 21:7447–7462.

Checklist

1. For all models and algorithms presented, check if you include:

   (a) A clear description of the mathematical setting, assumptions, algorithm, and/or model. [Yes]

(b) An analysis of the properties and complexity (time, space, sample size) of any algorithm. [Yes]

(c) (Optional) Anonymized source code, with specification of all dependencies, including external libraries. [No]

2. For any theoretical claim, check if you include:

   (a) Statements of the full set of assumptions of all theoretical results. [Not Applicable]

   (b) Complete proofs of all theoretical results. [Not Applicable]

   (c) Clear explanations of any assumptions. [Not Applicable]

3. For all figures and tables that present empirical results, check if you include:

   (a) The code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL). [Yes]

   (b) All the training details (e.g. data splits, hyperparameters, how they were chosen). [Yes]

   (c) A clear definition of the specific measure or statistics and error bars (e.g. with respect to the random seed after running experiments multiple times). [Yes]

   (d) A description of the computing infrastructure used. (e.g. type of GPUs, internal cluster, or cloud provider). [Yes]

4. If you are using existing assets (e.g. code, data, models) or curating/releasing new assets, check if you include:

   (a) Citations of the creator If your work uses existing assets. [Not Applicable]

   (b) The license information of the assets, if applicable. [Not Applicable]

   (c) New assets either in the supplemental material or as a URL, if applicable. [Not Applicable]

   (d) Information about consent from data providers/curators. [Not Applicable]

   (e) Discussion of sensible content if applicable, e.g. personally identifiable information or offensive content. [Not Applicable]

5. If you used crowdsourcing or conducted research with human subjects, check if you include:

   (a) The full text of instructions given to participants and screenshots. [Not Applicable]

   (b) Descriptions of potential participant risks, with links to Institutional Review Board (IRB) approvals if applicable. [Not Applicable]

   (c) The estimated hourly wage paid to participants and the total amount spent on participant compensation. [Not Applicable]

Conformal Prediction in Hierarchical Classification with Constrained Representation Complexity: Supplementary Materials

A EXPERIMENTAL SETUP

We use a MobileNetV2 convolutional neural network (Sandler et al., 2018) pretrained on ImageNet (Krizhevsky et al., 2017), in order to obtain hidden representations for all image datasets. The cross-entropy loss is minimized using the Adam optimizer, with a learning rate of 1×10^{-5} and momentum set to 0.99. We set the number of epochs to 2 and 20, for the Caltech and other datasets, respectively. We train all models end-to-end on a GPU, by using the PyTorch library (Paszke et al., 2019) and infrastructure with the following specifications:

- CPU: Intel i7-6800K 3.4 GHz (3.8 GHz Turbo Boost)
- GPU: NVIDIA GTX 1080 Ti 11GB
- RAM: 64GB DDR4-2666

For the AMB and DBpedia dataset, we use the same model and training procedure as described in Theunissen et al. (2024) and Mortier et al. (2021), respectively.

B ADDITIONAL RESULTS

In this section, we include additional results to those presented in Table 2, namely, results for (N)CRSVP-1 and (N)CRSVP-2. In addition, we give insights into potential conditional coverage violations by examining the Pearson correlation between coverage and prediction set size. A strong correlation may indicate a potential violation of conditional coverage (Feldman et al., 2021). While this metric serves as a useful proxy for evaluating conditional coverage, it is not definitive, as a correlation of zero does not necessarily guarantee conditional coverage, as shown in Rossellini et al. (2024). The results are summarized in Table C3, C4 and C5.

First, note that CRSVP is different from CRSVP-1. The restriction is the same, however, the set of solutions for both algorithms is different: For CRSVP, starting from the mode of the distribution, the solution space is given by the mode and its ancestors. For CRSVP-1, the solution space is given by line 7 and 6 in Algorithm 2 and 3, respectively, namely, the most common ancestor (8) for the top-1, top-2, top-3, etc., classes. In most cases, we observe low correlation for (N)CRSVP and (N)CRSVP-*r*, reflecting strong performance regarding conditional coverage.

Finally, in Figure C4, we visually explore the relationship between representation complexity and set size for the unrestricted set-valued predictors (LAC, APS) and restricted set-valued predictors (CRSVP, CRSVP-*r*). As is clear from both Table C3, C4, C5 and Figure C4, increasing the representation complexity results in better efficiency.

Table C3: Additional results for CIFAR-10 and AMB. Coverage, efficiency, representation complexity and Pearson correlation values for the following unrestricted set-valued predictors: LAC, NPS, APS, and restricted set-valued predictors: NCRSVP, CRSVP, NCRSVP-*r*, and CRSVP-*r*. The confidence level is set to 90%, and calibration and test sets are resampled 10 times.

DATASET	CIFAR-10				AMB			
ALG.	COV.	SIZE	REPR. COMP.	CORR.	COV.	SIZE	REPR. COMP.	CORR.
LAC	0.899 ± 0.005	1.473 ± 0.023	1.451 ± 0.021	-0.07 ± 0.011	0.899 ± 0.010	1.128 ± 0.020	1.132 ± 0.019	0.060 ± 0.025
NPS	0.997 ± 0.001	5.125 ± 0.058	3.552 ± 0.017	0.016 ± 0.010	1.000 ± 0.000	23.98 ± 1.195	17.38 ± 0.702	-
APS	0.899 ± 0.003	1.849 ± 0.019	1.824 ± 0.015	0.111 ± 0.014	0.900 ± 0.009	1.685 ± 0.055	1.776 ± 0.051	0.177 ± 0.015
NCRSVP	1.000 ± 0.000	10.00 ± 0.000	1.000 ± 0.000	-	1.000 ± 0.000	50.85 ± 1.304	1.000 ± 0.000	-
CRSVP	0.899 ± 0.005	3.899 ± 0.049	1.000 ± 0.000	0.134 ± 0.009	0.900 ± 0.009	4.856 ± 0.261	1.000 ± 0.000	0.137 ± 0.008
NCRSVP-1	1.000 ± 0.000	10.00 ± 0.000	1.000 ± 0.000	-	1.000 ± 0.000	51.13 ± 1.338	1.000 ± 0.000	-
CRSVP-1	0.898 ± 0.005	3.580 ± 0.050	1.000 ± 0.000	0.207 ± 0.005	0.900 ± 0.008	4.890 ± 0.266	1.000 ± 0.000	0.148 ± 0.007
NCRSVP-2	0.998 ± 0.001	6.775 ± 0.047	1.616 ± 0.004	0.028 ± 0.009	1.000 ± 0.000	43.35 ± 1.578	1.511 ± 0.011	-
CRSVP-2	0.899 ± 0.005	2.279 ± 0.036	1.457 ± 0.005	0.105 ± 0.012	0.900 ± 0.009	2.666 ± 0.117	1.248 ± 0.011	0.124 ± 0.007
NCRSVP-3	0.997 ± 0.001	5.861 ± 0.064	2.368 ± 0.008	0.024 ± 0.009	1.000 ± 0.000	39.26 ± 1.574	1.829 ± 0.009	-
CRSVP-3	0.899 ± 0.003	1.946 ± 0.025	1.691 ± 0.009	0.103 ± 0.014	0.899 ± 0.009	2.184 ± 0.063	1.394 ± 0.018	0.114 ± 0.009

Table C4: Additional results for Caltech-101 and DBpedia. Coverage, efficiency, representation complexity and Pearson correlation values for the following unrestricted set-valued predictors: LAC, NPS, APS, and restricted set-valued predictors: NCRSVP, CRSVP, NCRSVP-*r*, and CRSVP-*r*. The confidence level is set to 90%, and calibration and test sets are resampled 10 times.

DATASET	CALTECH-101				DBPEDIA			
ALG.	COV.	SIZE	REPR. COMP.	CORR.	COV.	SIZE	REPR. COMP.	CORR.
LAC	0.900 ± 0.007	0.920 ± 0.008	1.000 ± 0.000	0.886 ± 0.021	0.899 ± 0.001	0.931 ± 0.002	1.000 ± 0.000	0.814 ± 0.005
NPS	1.000 ± 0.000	96.54 ± 0.065	1.454 ± 0.072	-	1.000 ± 0.000	59.17 ± 0.272	53.18 ± 0.257	-0.010 ± 0.003
APS	0.900 ± 0.006	1.165 ± 0.015	1.251 ± 0.015	0.184 ± 0.013	0.901 ± 0.002	11.33 ± 0.287	11.90 ± 0.277	0.129 ± 0.001
NCRSVP	1.000 ± 0.000	96.25 ± 0.049	1.000 ± 0.000	-	0.999 ± 0.000	162.9 ± 0.492	1.000 ± 0.000	0.019 ± 0.001
CRSVP	0.901 ± 0.005	4.400 ± 0.254	1.000 ± 0.000	0.062 ± 0.009	0.901 ± 0.002	26.62 ± 0.417	1.000 ± 0.000	0.087 ± 0.002
NCRSVP-1	1.000 ± 0.000	70.36 ± 0.550	1.000 ± 0.000	-	1.000 ± 0.000	183.2 ± 0.380	1.000 ± 0.000	0.023 ± 0.003
CRSVP-1	0.900 ± 0.005	4.173 ± 0.241	1.000 ± 0.000	0.079 ± 0.006	0.901 ± 0.002	25.59 ± 0.386	1.000 ± 0.000	0.119 ± 0.002
NCRSVP-2	1.000 ± 0.000	66.39 ± 0.528	1.115 ± 0.004	-	0.999 ± 0.000	150.6 ± 0.418	1.351 ± 0.002	0.013 ± 0.001
CRSVP-2	0.900 ± 0.007	2.381 ± 0.110	1.085 ± 0.004	0.054 ± 0.014	0.901 ± 0.002	19.34 ± 0.484	1.191 ± 0.002	0.107 ± 0.001
NCRSVP-3	1.000 ± 0.000	63.61 ± 0.517	1.306 ± 0.008	-	0.999 ± 0.000	127.9 ± 0.501	1.985 ± 0.004	0.004 ± 0.001
CRSVP-3	0.901 ± 0.006	1.784 ± 0.086	1.133 ± 0.007	0.066 ± 0.011	0.901 ± 0.001	17.71 ± 0.423	1.380 ± 0.005	0.113 ± 0.001

Table C5: Additional results for Caltech-256 and PlantCLEF 2015. Coverage, efficiency, representation complexity and Pearson correlation values for the following unrestricted set-valued predictors: LAC, NPS, APS, and restricted set-valued predictors: NCRSVP, CRSVP, NCRSVP-*r*, and CRSVP-*r*. The confidence level is set to 90%, and calibration and test sets are resampled 10 times.

DATASET	CALTECH-256				PLANTCLEF 2015			
ALG.	COV.	SIZE	REPR. COMP.	CORR.	COV.	SIZE	REPR. COMP.	CORR.
LAC	0.900 ± 0.004	1.931 ± 0.040	1.926 ± 0.040	-0.29 ± 0.006	0.899 ± 0.003	25.50 ± 0.450	24.33 ± 0.419	-0.20 ± 0.009
NPS	0.999 ± 0.000	62.25 ± 1.330	50.12 ± 0.996	-0.01 ± 0.009	0.970 ± 0.001	123.6 ± 2.026	104.4 ± 1.560	0.003 ± 0.006
APS	0.901 ± 0.004	3.640 ± 0.099	3.680 ± 0.098	-0.05 ± 0.008	0.899 ± 0.003	44.12 ± 0.994	40.19 ± 0.836	-0.01 ± 0.010
NCRSVP	0.999 ± 0.000	208.4 ± 1.780	1.000 ± 0.000	-	1.000 ± 0.000	998.9 ± 0.224	1.000 ± 0.000	-
CRSVP	0.900 ± 0.003	44.69 ± 1.252	1.000 ± 0.000	0.100 ± 0.007	0.900 ± 0.002	520.9 ± 4.745	1.000 ± 0.000	0.340 ± 0.004
NCRSVP-1	0.999 ± 0.000	215.5 ± 1.885	1.000 ± 0.000	-	1.000 ± 0.000	998.9 ± 0.176	1.000 ± 0.000	-
CRSVP-1	0.901 ± 0.003	44.24 ± 1.297	1.000 ± 0.000	0.139 ± 0.004	0.900 ± 0.003	534.5 ± 4.492	1.000 ± 0.000	0.356 ± 0.005
NCRSVP-2	0.999 ± 0.000	179.0 ± 2.035	1.242 ± 0.004	-	1.000 ± 0.000	997.0 ± 0.443	1.002 ± 0.001	-
CRSVP-2	0.901 ± 0.003	27.63 ± 0.898	1.270 ± 0.006	0.073 ± 0.007	0.898 ± 0.003	442.1 ± 4.337	1.290 ± 0.004	0.295 ± 0.004
NCRSVP-3	1.000 ± 0.000	163.6 ± 2.070	1.632 ± 0.005	-	1.000 ± 0.000	995.3 ± 0.492	1.006 ± 0.001	-
CRSVP-3	0.901 ± 0.003	20.30 ± 0.830	1.498 ± 0.009	0.045 ± 0.009	0.899 ± 0.004	389.7 ± 5.898	1.632 ± 0.010	0.262 ± 0.004

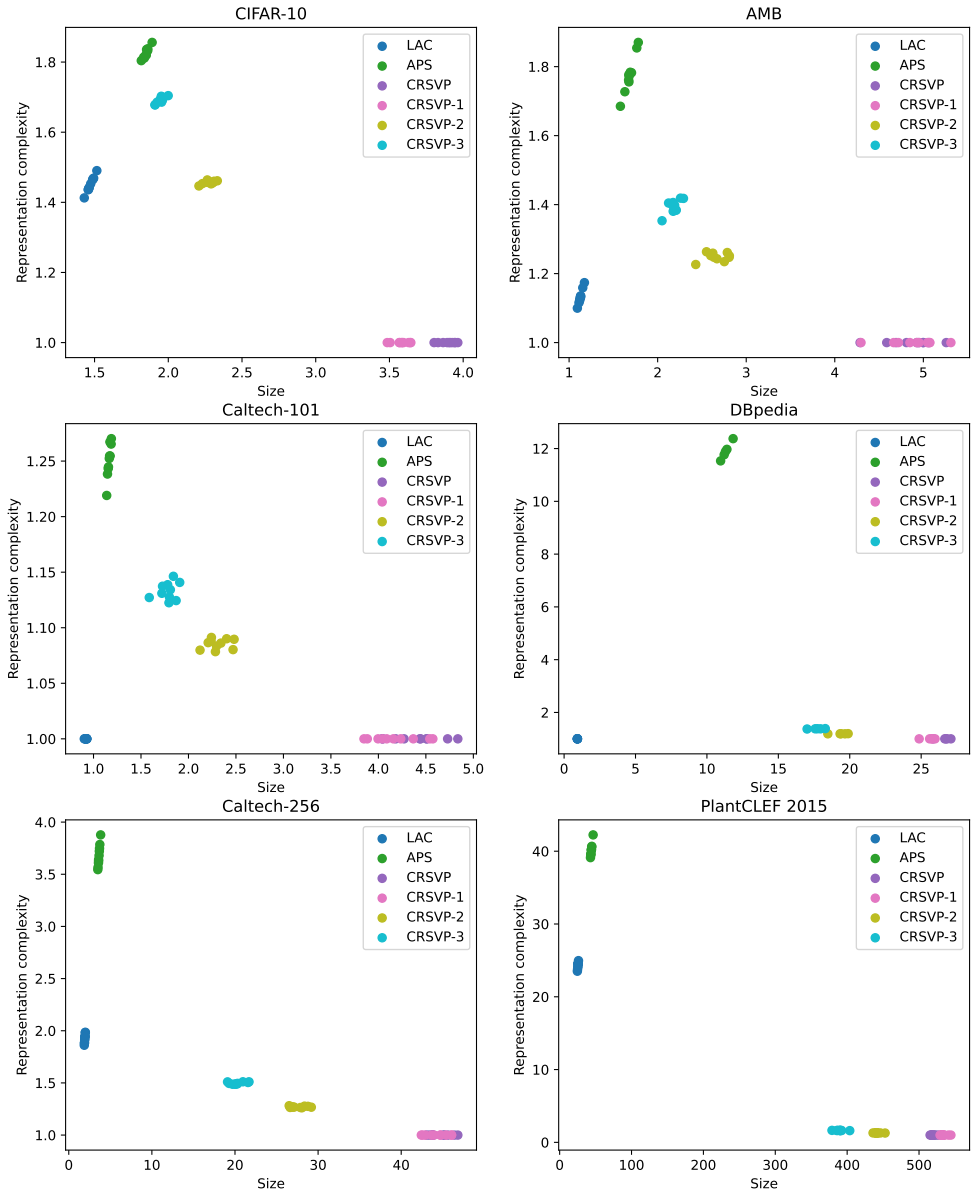

Figure C4: Representation complexity versus set size for the following unrestricted set-valued predictors: LAC and APS, and restricted set-valued predictors: CRSVP and CRSVP-*r*, for all datasets. The confidence level is set to 90%, and calibration and test sets are resampled 10 times.