Controlling Information Leakage in Concept Bottleneck Models with Trees

Anonymous authors
Paper under double-blind review

Abstract

As AI models grow larger, the demand for accountability and interpretability has become increasingly critical for understanding their decision-making processes. Concept Bottleneck Models (CBMs) have gained attention for enhancing interpretability by mapping inputs to intermediate concepts before making final predictions. However, CBMs often suffer from information leakage, where additional input data, not captured by the concepts, is used to improve task performance, complicating the interpretation of downstream predictions. In this paper, we introduce a novel approach for training both joint and sequential CBMs that allows us to identify and control leakage using decision trees. Our method quantifies leakage by comparing the decision paths of hard CBMs with their soft, leaky counterparts. Specifically, we show that soft leaky CBMs extend the decision paths of hard CBMs, particularly in cases where concept information is incomplete. Using this insight, we develop a technique to better inspect and manage leakage, isolating the subsets of data most affected by this. Through synthetic and real-world experiments, we demonstrate that controlling leakage in this way not only improves task accuracy but also yields more informative and transparent explanations.

1 Introduction

Deep learning models have demonstrated significant capabilities and been widely adopted in applications such as image recognition, natural language processing, and disease prediction. However, their use in high-stakes fields like healthcare, pollution monitoring, credit risk, and criminal justice has raised concerns about transparency and flawed predictions (Hu et al., 2019). While numerous methods provide post-hoc analysis of trained neural networks (e.g., Ghorbani et al. (2019); Zhou et al. (2018)), these explanations do not always align with human understanding (Rudin, 2019).

Several recent studies suggest explicitly aligning intermediate outputs of neural network models with predefined expert concepts during supervised training processes (e.g Koh et al. (2020); Chen et al. (2020); Kumar et al. (2009); Lampert et al. (2009)) through the use of Concept Bottleneck Models (CBMs). Given a high-dimensional input of features (such as the raw pixels of an image), CBMs first predict a set of human-understandable concepts, which are then used to predict the final task labels with the help of an interpretable label predictor. Thus, a user is able to follow the decision-making process of the label predictor, while the task performance can remain close to that of the black-box model (Koh et al., 2020).

Despite these characteristics, CBMs frequently suffer from *information leakage* (Mahinpei et al., 2021; Margeloiu et al., 2021). This occurs where unintentional signal from the data not captured by the concepts is used for predicting the label and potentially increases its accuracy. If the label predictions rely on this information, explanations derived from the concepts may be inaccurate and potentially misleading. Crucially, leakage compromises our ability to intervene on concepts – a key benefit of CBMs. Recent works have attempted to mitigate leakage in CBMs by allowing a residual layer or side channel to capture a set of unknown, *latent concepts* (e.g. Havasi et al. (2022a); Shang et al. (2024); Zabounidis et al. (2023); Heidemann et al. (2023); Vandenhirtz et al. (2024)). Yet, while leakage is reduced, the information captured by these latent concepts is difficult to interpret and not necessarily disentangled from the known concepts, only partially resolving the issue.

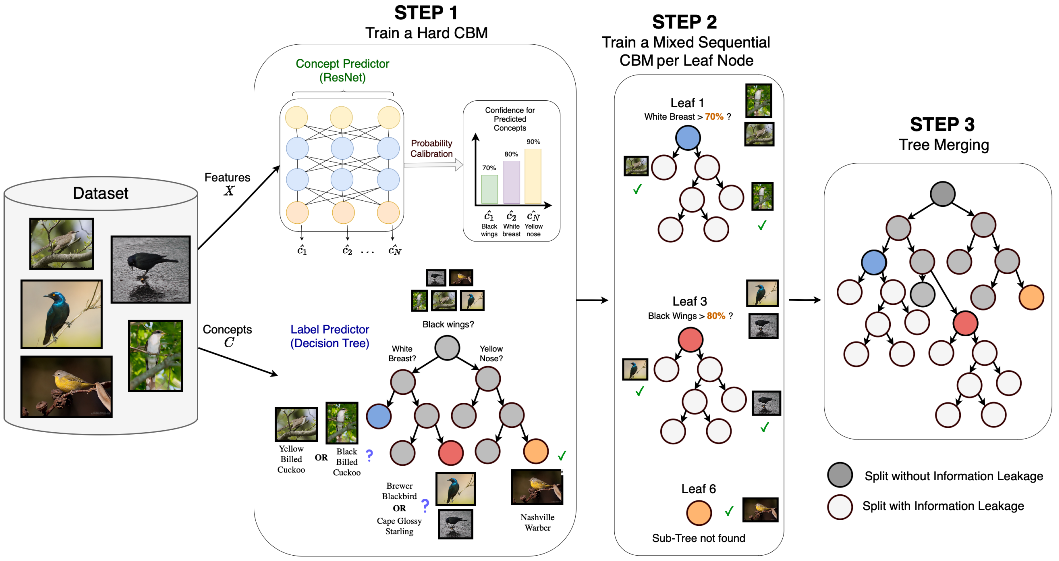


Figure 1: An overview of the Mixed CBM Algorithm (MCBM). *Step 1*: Two independent networks, a concept predictor with calibrated probability outputs and a decision tree label predictor (global tree) are trained. *Step 2*: A Sequential CBM (sub-tree) with mixed concept representations (*Mixed CBM*) further splits the leaf nodes of the global tree that present missing concept information and are prone to Leakage. *Step 3*: All trees are merged for global Leakage Inspection.

Given the current limitations, the purpose of this work is provide an interpretable method for inspecting where leakage occurs in CBMs and controlling this leakage using decision trees. We develop a method called Mixed CBM Training with Trees (MCBM) for better inspecting and managing leakage in those subsets, and introduce both sequential (MCBM-Seq) and joint training (MCBM-Joint) variants of this method for different scenarios of incomplete concept sets. An overview of our architecture can be found in Figure 1. Due to their hierarchical nature, trees allow us to first identify leaf nodes with subsets that are missing concept information, and subsequently control information leakage to only specialise those decision paths. This is achieved through a 3-step process, where a Hard CBM (global tree) is first trained, then an individual sub-tree is trained to extend each leaf node only if it can take advantage of leakage, and finally all trees are merged for global inspection.

Our contributions are as follows: We introduce a tree-based method for inspecting and controlling leakage in CBMs. We show that our method enables more interpretable decision-making with explanations that have higher accuracy and are guaranteed to make faithful predictions (in terms of fidelity) when concept sets are incomplete. We also demonstrate that our method allows us to quantify leakage for specific data subsets associated with tree regions where concept information may be incomplete, while identifying those decision rules most affected by leakage. Finally, we show that the derived group-based explanations can be very meaningful in real-life decision-making scenarios, and provide practical recommendations for selecting the appropriate method for training a CBM, based on the context of a problem and preferences of the user.

2 Related Work

Concept Bottleneck Models and Information Leakage. Concept Bottleneck Models (CBMs) (Koh et al., 2020; Lampert et al., 2009; Kumar et al., 2009) are trained on data with covariates $x \in X$, target $y \in Y$, and annotated binary concepts $c \in C$. These models use a neural network f_θ, parameterized by θ and structured as $\langle g_\psi, h_\phi \rangle$ (Leino et al., 2018), to enforce a concept bottleneck $\hat{c} = h_\phi(x)$. The final output depends solely on the predicted concepts $\hat{c}$. Soft CBMs (Chen et al., 2020; Koh et al., 2020) improve prediction by using probabilistic concept values but are prone to information leakage from the concept predictor to the label predictor (Margeloiu et al., 2021; Mahinpei et al., 2021). Most CBM research focuses on extending concept representations in the embedding space to enhance predictive power (Zarlenga et al., 2024; Oikarinen et al., 2023; Kim et al., 2023; Semenov et al., 2024), while neglecting information leakage. Some works address leakage by allowing missing concept information to bypass concept representations through a Residual Layer (Yuksekgonul et al., 2023; Shang et al., 2024). Havasi et al. (2022b) tackle missing information with a side channel and an auto-regressive concept predictor, but these approaches struggle with interpretability and disentanglement of residual information (Zabounidis et al., 2023). Another approach by Marconato et al. (2022) rejects test samples prone to leakage, though it relies on assumptions about the prior distribution. Unlike these approaches, the work we present here enables both interpretability and leakage inspection, without any prohibitive assumptions on the distribution that may not hold in practice.

Explainable Label Predictors. According to CBM definitions (Koh et al., 2020), any interpretable machine learning model can serve as a label predictor, such as Logistic Regression (McKelvey & Zavoina, 1975), Generalised Additive Models (Hastie & Tibshirani, 1985), or Decision Trees (Breiman et al., 1984; Kass, 1980). Recent work integrates neural networks with interpretable decision-making. Wu et al. (2018; 2020) introduce tree-regularization to approximate neural network boundaries with decision trees. Ciravegna et al. (2021) propose the ψ network for logic-based concept explanations using L1-regularization and pruning to extract interpretable First Order Logic formulas. Barbiero et al. (2022) enhance this with the Entropy-Net, using Entropy Loss for more concise logic formulas. Ghosh et al. (2023) further introduce a mixture of Entropy-Net experts for specialized explanations while maintaining performance. Yet, these concept-based explanations assume concept probabilities without evaluating potential label information leakage as we propose.

3 PRELIMINARIES

Problem Setting. We consider a classification task with $N = \{1, ..., n\}$ samples, $K = \{1, ..., k\}$ concepts and $R = \{1, ..., r\}$ classes. We assume a training set $D_{train} = \left\{\left(x^{(i)}, c^{(i)}, y^{(i)}\right)\right\}_{i=1}^{N}$, where: $x^{(i)}$ is an input feature vector (e.g. an image) of the input space $X \subset \mathbb{R}^d$; $c^{(i)}$ is a categorical vector of k concepts of the concept space $C \subset \{0,1\}^k$; $y^{(i)}$ is a one-hot encoded vector of the target space $Y \subset \{0,1\}^r$. A test set N_{test} with $X_{test} \subset \mathbb{R}^d$ is also given, without annotated concepts.

Concept Bottleneck Models. The architecture of a CBM first introduces a *concept predictor* $f(W_1) : X \rightarrow C$ that maps inputs to concepts. It then uses a *label predictor* $g(W_2) : C \rightarrow Y$ that maps concepts to targets. This is typically any interpretable model, such that the relationship from concepts to targets can be explained e.g linear layer decision trees. CBMs can be classified into two categories (Havasi et al., 2022a; Koh et al., 2020):

Hard CBMs: At test time, the label predictor only accepts binary (hard) concepts as inputs. The networks f and g are trained independently on the ground truth data.

$$L_C = L_{W_1}(\hat{C}, C) = \sum_i L_{W_1}\left(f(x^{(i)})\,;\,c^{(i)}\right) = \sum_{i,k} L_{W_1}\left(f(x^{(i)})[\,k\,]\,;\,c^{(i)}[\,k\,]\right), k \in K \tag{1}$$

$$L_Y = L_{W_2}(\hat{Y}, Y) = \sum_i L_{W_2}\left(g(c^{(i)})\,;\,y^{(i)}\right) \tag{2}$$

We use cross-entropy loss as the task loss L_Y. For the concept loss L_C, we use the sum of binary cross-entropy losses for independent concepts, with the sum of cross-entropy losses for groups of mutually-exclusive concepts. At test time, we make a prediction for a sample x_* by first converting the predicted logits into concept probabilities $\hat{c} = \sigma(f(x_*))$, where σ is either a sigmoid function for independent concepts or a softmax function for mutually-exclusive concepts. We convert the probabilities to binary representations $\hat{c}_{bin}$ either through thresholding (sigmoid) or the *argmax* operator (softmax), and we pass them to the label predictor: $y_* = g(\hat{c}_{bin})$.

Soft CBMs. At test time, the label predictor accepts concept probabilities (soft concepts) as inputs. These can be trained either independently, sequentially or jointly. Independent training uses a procedure identical to Hard CBMs, based on Eq. 1, 2. At test time, we use the concept probabilities: $y_* = g(\hat{c})$, where $\hat{c} = \sigma(f(x_*))$. For sequential training, the network f is trained according to the objective of Eq. 1, and then the predicted concepts $\hat{c}^{(i)} = \hat{g}(x^{(i)})$ are used as input to train the network g, minimising the loss: $L_Y = L_{W_2}(\hat{Y}, Y) = \sum_i L_{W_2}\left(g(\hat{c}^{(i)})\,;\,y^{(i)}\right)$, where $\hat{c}^{(i)} =$

$\sigma(f(x^{(i)}))$. Joint training trains both networks simultaneously. The hyper-parameter λ_C controls the relative importance of the two tasks. Assuming $\hat{c}^{(i)} = \sigma(f(x^{(i)}))$:

$$L = L_Y + \lambda_C L_C = \sum_i L_{W_1,W_2}\left(g(\hat{c}^{(i)})\,;\,y^{(i)}\right) + \lambda_C \sum_i L_{W_1}\left(f(x^{(i)})\,;\,c^{(i)}\right) \tag{3}$$

Leakage in CBMs. While information leakage is defined by Mahinpei et al. (2021) as a type of unintended information that the concept representations capture, in this work we propose a more explicit definition:

Definition 3.1 *The amount of unintended information that is used to predict label y with soft concepts $\hat{c}$ that is not present in hard representation c.*

In our work, we measure this as the *mutual information* of targets y and soft concepts $\hat{c}$ given the hard concepts c,

$$I_{\text{Leakage}} \;=\; I(y;\hat{c}\,|c) = H(y|c) - H(y|\hat{c},c) \tag{4}$$

4 Tree-Based Leakage Inspection and Control

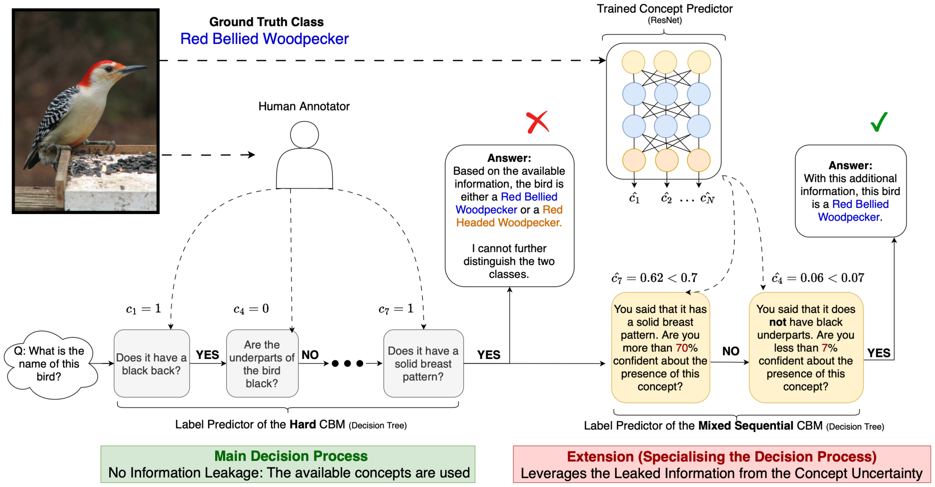
In what follows, we present the core contribution of our work. Specifically, given both a trained hard CBM and a trained (either sequentially or jointly) soft CBM, we would like to determine whether it is possible to inspect any information leakage that the soft model exploits to make its predictions, and understand how this information was used. Based on Definition 3.1, we know that any additional information used to predict y that is not contained in concepts c, should be captured by the difference of conditional mutual information terms in Eq. 4. A naive attempt to answer this question is thus to first train the same concept predictor for both hard and soft models, and then use separate decision tree classifiers for each CBM as label predictors and subsequently inspect their trees to observe how they differ. However, inspecting the corresponding decision tree label predictors for both hard and soft CBMs is non-trivial as the trees be incomparable and contain very different splits to distinguish samples (an example is given in Appendix A.3).

Motivated by this challenge, we present a new training method with *mixed concept representations.* Our key insight is that any subset of data that cannot be further split by a hard CBM's decision tree but can be split by a soft leaky CBM's tree is vulnerable to information leakage. The method has three steps, shown in Fig. 1. First, we train a *hard, leakage-free* CBM with a decision tree as the *global* label predictor. This tree is decomposed into decision paths with some concepts, while other concepts are predicted independently with calibrated probabilities. In the second step, the decision paths are used to train a *mixed CBM*, combining hard and soft concepts with a new tree-based label predictor. Finally, the global tree and sub-trees are merged for complete inspection.

4.1 The Mixed Sequential CBM (MCBM-Seq) Training Algorithm.

Identifying samples corresponding to certain concepts. We train the concept predictor $f : X \rightarrow C$ according to Eq. 1 and the label predictor $g : C \rightarrow Y$ independently. In principle, the concept predictor could be any deep learning model; we replace the label predictor of a standard CBM by a decision tree (Breiman et al., 1984; Kass, 1980). Specifically, the tree is constrained to a minimum number of samples per leaf (msl) to prevent overfitting. We refer to this tree as the "global" tree containing decision paths with all the concepts. We then decompose the tree into a set of decision paths $T_{hard} = \{P_1, ..., P_M\}$, where M is the number of leaf nodes. Each decision path corresponds to a set of binary decision rules that are not affected by leakage, since the tree was trained on the ground truth binary concepts. Fig. 2 depicts these rules intuitively as a set of True or False questions used to distinguish a test sample from other classes.

Calibrating the concept probabilities to prevent overconfident predictions. To overcome the problem of overconfidence in the predictions of deep neural networks (Guo et al., 2017), we calibrate the predicted concept probabilities of the trained concept predictor. We perform Platt scaling (PLATT, 1999) for binary (independent) concept predictions, and temperature scaling for multi-class (mutually-exclusive) concept predictions (Guo et al., 2017). We argue that the calibration step is crucial for the interpretability of Sequential CBMs (which are trained in the next step), since the


Ground Truth Class
Red Bellied Woodpecker
Human Annotator
Trained Concept Predictor
(ResNet)
$\hat{c_1}$ $\hat{c_2}$ … $\hat{c_N}$
$c_1 = 1$
$c_4 = 0$
$c_7 = 1$
YES
NO
YES
$\hat{c_7} = 0.62 < 0.7$
$\hat{c_4} = 0.06 < 0.07$
NO
YES
Label Predictor of the **Hard** CBM (Decision Tree)
Label Predictor of the **Mixed Sequential** CBM (Decision Tree)

Q: What is the name of this bird?

Does it have a black back?

Are the underparts of the bird black?

Does it have a solid breast pattern?

Answer: Based on the available information, the bird is either a Red Bellied Woodpecker or a Red Headed Woodpecker.

I cannot further distinguish the two classes.

You said that it has a solid breast pattern. Are you more than 70% confident about the presence of this concept?

You said that it does **not** have black underparts. Are you less than 7% confident about the presence of this concept?

Answer: With this additional information, this bird is a Red Bellied Woodpecker.

Main Decision Process
No Information Leakage: The available concepts are used

Extension (Specialising the Decision Process)
Leverages the Leaked Information from the Concept Uncertainty

Figure 2: Summary of the Decision Making Process of The Mixed Sequential CBM Algorithm (MCBM-Seq) when classifying an image with annotated concepts. The process is described intuitively as a conversation between the Tree label predictor and two entities that provide the input concepts: the Human Annotator and the Concept Predictor. The concept probabilities are used to specialise the decision process only when the available annotated concepts are not sufficient.

decision rules of the label predictor are based on the true concept probabilities and can often be human intuitive (see section A.10).

Training a mixed sequential CBM for each decision path. We first isolate the set of all concepts $K_m \subseteq K$ used in the decision splits of each path P_m, and the set of the remaining concepts $K'_m = K - K_m$ not used in the path. Then, for each decision path in the global tree, we train a mixed sequential CBM. Specifically, we extract a subset of the dataset $\{X_m, C_m, Y_m\}$ with the training samples classified by the decision path P_m. We use the already trained network $f : X \rightarrow C$ as the concept predictor of the CBM. Then for each sample i, we construct a new concept vector c_i^* as follows: For the concepts appearing in the decision path, we assign the calibrated soft probability given by the concept predictor: $c_i^*\,[k] = \hat{c}_i\,[k] = f(x_i)[k]$, $\forall\, k \in K_m$. For the remaining concepts, we assign the hard (binary) value: $c_i^*\,[k] = c_i\,[k]$, $\forall\, k \in K'_m$. Since each concept vector does not contain exclusively hard (binary) concept values or soft concept probabilities, but a combination of both, we name this architecture a *Mixed CBM*. Next, we train a new Decision Tree as the label predictor of the CBM, however constrained on the *same number of minimum samples per leaf as the global tree (msl)*. This ensures that the new sub-tree, if found, can further specialise the data *only* because of the additional leaky information encoded in the concept probabilities, and not because of a smaller constraint in the number of samples allowed per leaf.

We should emphasise that the concept predictor is only trained *once*, and thus our method does not introduce any computational overhead compared to a Sequential CBM with a single decision tree as label predictor (see section A.4). A second important detail is that we train a CBM with *mixed* concept representations per leaf subset, and not a purely Soft Sequential CBM. This allows us to investigate if the soft representation of one or more of the concepts *that have appeared in the decision path of this subset in the global tree* could further specialise the decision process. This is crucial in order to quantify leakage in section 4.2 and to approximate our definition of leakage in Eq. 4, because only the concepts K_m present in the non-leaky decision path of a leaf m in the global tree are those shared by all samples s in the leaf, and thus satisfy the definition of the conditional entropy for this leaf: $H(y_s|c_k), \forall k \in K_m$. Instead, if we trained a purely Soft CBM, some of the concepts may not be shared by this group. Referring again to the example of Fig. 2, we would like to observe how a decision rule based on the confidence of the concept predictor (shown in a yellow box) can specialise one or more binary rules that appeared in the global tree (shown in a white box).

Merging the sub-trees to the global tree. This final step is optional and only used when the size of the tree is reasonable and the user would like to grasp a global picture of label classifications

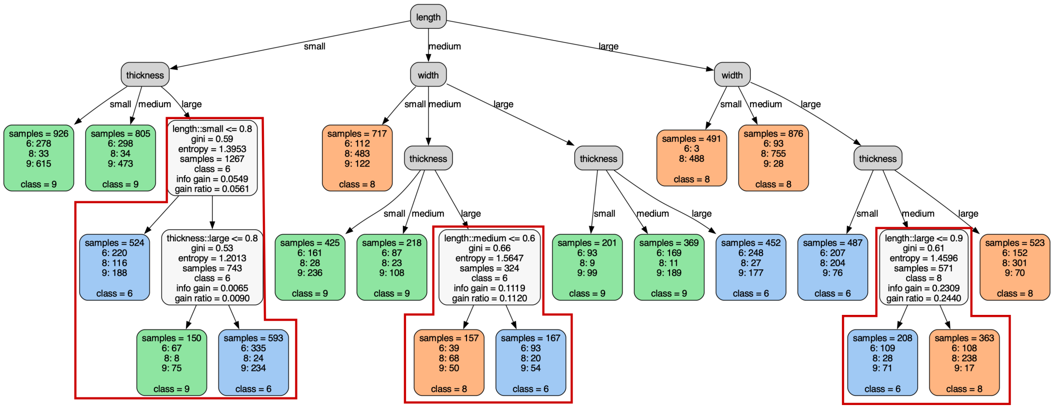

length
small
medium
large
thickness
width
width
small
medium
large
samples = 926 6: 278 8: 33 9: 615 class = 9
samples = 805 6: 298 8: 34 9: 473 class = 9
length::small <= 0.8 gini = 0.59 entropy = 1.3953 samples = 1267 class = 6 info gain = 0.0549 gain ratio = 0.0561
samples = 717 6: 112 8: 483 9: 122 class = 8
thickness
thickness
samples = 491 6: 3 8: 488 class = 8
samples = 876 6: 93 8: 755 9: 28 class = 8
thickness
samples = 524 6: 220 8: 116 9: 188 class = 6
thickness::large <= 0.8 gini = 0.53 entropy = 1.2013 samples = 743 class = 6 info gain = 0.0065 gain ratio = 0.0090
samples = 425 6: 161 8: 28 9: 236 class = 9
samples = 218 6: 87 8: 23 9: 108 class = 9
length::medium <= 0.6 gini = 0.66 entropy = 1.5647 samples = 324 class = 6 info gain = 0.1119 gain ratio = 0.1120
samples = 201 6: 93 8: 9 9: 99 class = 9
samples = 369 6: 169 8: 11 9: 189 class = 9
samples = 452 6: 248 8: 27 9: 177 class = 6
samples = 487 6: 207 8: 204 9: 76 class = 6
length::large <= 0.9 gini = 0.61 entropy = 1.4596 samples = 571 class = 8 info gain = 0.2309 gain ratio = 0.2440
samples = 523 6: 152 8: 301 9: 70 class = 8
samples = 150 6: 67 8: 8 9: 75 class = 9
samples = 593 6: 335 8: 24 9: 234 class = 6
samples = 157 6: 39 8: 68 9: 50 class = 8
samples = 167 6: 93 8: 20 9: 54 class = 6
samples = 208 6: 109 8: 28 9: 71 class = 6
samples = 363 6: 108 8: 238 9: 17 class = 8

Figure 3: The MCBM-Seq algorithm for a reduced Morpho-MNIST dataset with digits 6, 8 and 9 and concepts “length”, “thickness” and “width”. The final tree merged the sub-trees is shown. If a sub-tree is found, it replaces the leaf node of the hard CBM and is highlighted in a red box. The remaining leaf nodes are unaffected by leakage. This architecture allows us to both inspect and restrict leakage only to subsets with missing concept information.

and Leakage. For group-specific explanations, an analysis per individual decision path can be more informative (see section 5.3). A visualisation of a merged tree after performing the algorithm is provided in Fig. 3. At *test time*, we retrieve the decision path that classifies each sample in the *global* tree, and make a prediction using the corresponding soft tree. The complete pseudo-code is provided in Algorithm 1.

4.2 Quantifying Information Leakage with Trees

The MCBM-Seq algorithm allows us to inspect and quantify Information Leakage. Consider a subset of samples s that end up in one of the leaf nodes of the global tree, and consider a soft concept $\hat{c}_k$ based on which we perform the first split in the Mixed Sequential CBM. Thus, the subset s is divided into two new subsets, s_1 and s_2, and the new tree has three nodes in total. We also assume the target distribution of subsets s, s_1 and s_2 are y_s, y_{s_1} and y_{s_2} respectively. We can use the Information Gain (Breiman et al., 1984) we achieve when splitting a node with a soft concept $\hat{c}_k$ as a measure of the Information Leakage that the soft concept provides, based on Eq. 4:

$$I_{\text{Leakage}}(\hat{c}_k) \;\approx\; H(y_s) - \left[\frac{|s_1|}{|s|}H(y_{s_1}) + \frac{|s_2|}{|s|}H(y_{s_2})\right] \;=\; IG(\hat{c}_k) \tag{5}$$

This is only valid because of the way we constructed our tree, by first using the hard concepts to fit the global tree and then specialising the leaf nodes with soft concept splits. Also, this formulation allows us to inspect and quantify leakage **specifically for each split**, by adjusting the concept values c_s and $\hat{c}_k$ accordingly. Refer to Appendix A.2 for more details.

4.3 Mixed Joint CBM Training with Trees (MCBM-Joint)

Joint CBMs tend to achieve higher task performance compared to the other forms of CBM Training, due to the end-to-end optimization procedure of Eq. 3 (Koh et al., 2020). However, this comes at the expense of interpretability. According to (Mahinpei et al., 2021), they are more prone to Information Leakage compared to Sequential CBMs because the label predictor can ”shape” the concept probabilities in an unexplainable way in order to improve the task performance while decreasing concept accuracy. In practice, this trade-off is controlled by the λ_C parameter of Eq. 3. Unlike Sequential CBMs, these probabilities do not correspond to the true confidence of the concept predictor, and thus concept-based explanations based on these probabilities are even less human-intuitive. However, we could control Information Leakage to only specialise the main decision rules, in order to achieve better interpretations when higher performance is a priority.

Algorithm 1 Mixed Sequential CBM Training (MCBM-Seq)

Input: N samples; K concepts; R classes; A dataset $D_{train} = \{X, C, Y\}$, where $X \subset \mathbb{R}^d$, $C \subset \{0,1\}^k$, $Y \subset \{0,1\}^r$; A set of N_{test} samples with $X_{test} \subset \mathbb{R}^d$; Minimum Samples per leaf (msl).

Output: A hard tree: T_{hard}; a set of soft trees, one for each leaf: $T_{soft} = \{T_1, ..., T_M\}$; test predictions $\hat{Y}$.

```
procedure TRAINING($D_{train}, msl$)
    Train the concept predictor $f : X \rightarrow \hat{C}$, where $\hat{C} \subset [0,1]^k$ (Eq. 1);
    Define a decision tree constrained on the min samples per leaf $T_{hard} = Tree\,(\,msl\,)$;
    Train the tree on the hard concept set: $T_{hard}.fit(\,C, Y\,)$;
    Decompose the tree into a set of $M$ decision paths $T_{hard} = \{P_1, ..., P_M\}$;
    Train a sub-Tree per leaf, by calling the procedure: $T_{soft}$ = TRAINING SUB-TREE($D_{train}, f, T_{hard}$);
    return $T_{hard}, T_{soft}$
end procedure

procedure TRAINING SUB-TREE($D_{train}, f, T_{hard}$)
    for each decision path $m \in M$ do
        Collect the data for the samples of $P_m$: $D_m = \{X_m, C_m, Y_m\}$, $X_m \subseteq X$, $C_m \subseteq C$, $Y_m \subseteq Y$;
        Get the concept probabilities for the samples of the path $\hat{C}_m = f(X_m)$;
        Calibrate the concept probabilities using Platt or Temperature scaling
        Isolate the set of concepts $K_m \subseteq K$ used as splits in the path $P_m$;
        Isolate the set of concepts $K'_m = K - K_m$ not used as splits in the path $P_m$;
        for each sample $i \in N_m$ do
            Initialise a new concept vector for the sample $c_i^*$;
            Assign the soft concept values of $i$ for the concepts used in the path: $c_i^*\,[k] = \hat{c}_i\,[k]$, $\forall\, k \in K_m$
            Assign the hard concept values of $i$ for the remaining concepts: $c_i^*\,[k] = c_i\,[k]$, $\forall\, k \in K'_m$
        end for
        Concatenate the concept vectors to create a new concept set for the path: $C_m^* = \left\{c_1^*, ..., c_{N_m}^*\right\}$;
        Define a decision tree using the same $msl$ constraint: $T_m = Tree\,(\,msl\,)$;
        Train the new tree: $T_m.fit\,(\,C_m^*, Y_m\,)$;
    end for
    return $T_{soft} = \{T_1, ..., T_M\}$
end procedure

procedure EVALUATION($X_{test}, f, T_{hard}, T_{soft}$)
    Predict the concept values $\hat{C} = f(X_{test})$;
    Get the decision paths for the test predictions $\{P_1, ..., P_M\} = T_{hard}.predict(\hat{C})$
    for each decision path $m \in M$ do
        Isolate the test samples of the path $N_m \subseteq N_{test}, \hat{C}_m \subseteq \hat{C}$
        Predict from the associated tree $\hat{Y}_m = T_m.predict(\hat{C}_m), T_m \in T_{soft}$
    end for
    return $\hat{Y} = \{\hat{Y}_1, ..., \hat{Y}_M\}$
end procedure
```

For this purpose, we propose the MCBM-Joint algorithm as a post-hoc analysis tool for trained Joint CBMs. The algorithm is identical with that of MCBM-Seq in section 4.1, but the concept probabilities are now extracted from the optimised concept predictor of the Joint CBM instead of an independently trained concept predictor. In contrast to MCBM-Seq, we do not calibrate these concept probabilities, since they do not correspond to the true confidence of the predictor. A visual intuition is shown in Appendix 7.

5 EXPERIMENTS

We evaluate the MCBM-Seq and MCBM-Joint methods in challenging image classification and medical settings, demonstrating the versatility of our approach across different metrics overall, as well as per decision path.

Baselines. We first compare our MCBM-Seq and MCBM-Joint methods with the standard modes of CBM training (Hard, Independent, Sequential) described in Koh et al. (2020), and we use a Decision Tree as the label predictor. For the MCBM-Joint method, we first optimise the concept encoder with the Joint training objective of Eq. 3 using a simple linear layer as the label encoder. We also train all CBM methods using the state-of-the-art Entropy-Net Barbiero et al. (2022) as the label-predictor, which achieves higher task accuracy compared to linear label predictors while also providing interpretable logic explanations.

Quantitative Metrics. In terms of performance, we evaluate our CBM methods using the a) *Task* and b) *Concept* Accuracy (Koh et al., 2020). To compare the interpretability and reliability of our methods compared to existing work, we use the following metrics defined by Barbiero et al. (2022): a) The *Explanation Accuracy* as the task performance of a method when using its extracted explanation formulas, and b) the *Fidelity of an Explanation* which measures how well each formula matches the model's predictions. Finally, we provide an estimate of Information Leakage as the Information Gain of leaky splits according Eq. 5, which is unique to our methods.

5.1 Datasets and Models

Morpho-MNIST (Castro et al., 2018): The dataset describes MNIST digits in terms of measurable shape attributes which we use as concepts. These are the thickness, area, length, width, height, and slant of digits. We categorise each real-valued concept in one of three equally spaced bins, indicating a ”small”, ”medium”, ”large” or value, resulting in mutually exclusive concept groups such as ”length::small”, ”length::medium” and ”length::large”. We use a LeNet model Lecun et al. (1998) as the concept predictor. Hyper-parameter details are given in Appendix A.7.

The CUB Dataset (Koh et al., 2020): The dataset consists of $n = 11,788$ pictures of 200 different bird species. There are 312 annotated binary concepts available. Similarly to Koh et al. (2020); Yeh et al. (2020), we only select those concepts that appear in at least 10% of the dataset, and thus we form a reduced set of 112 binary concepts. The data is denoised by converting instance-specific concepts to class-specific concepts via majority voting. To simulate a scenario of an incomplete concept set, we choose 45 concepts and denote the rest as missing. We train a ResNet-18 model (He et al., 2016) as the independently trained concept predictor (see Appendix A.6 and A.7 for pre-processing and hyperparameter tuning details).

MIMIC: MIMIC-IV (Medical Information Mart for Intensive Care IV) (Johnson et al., 2023) is a large, freely accessible dataset consisting of de-identified electronic health records from over 70,000 critical care patients. It includes data such as demographics, vital signs, laboratory results, medications, and clinical notes. Our binary classification task is to identify recovering or dying patients after ICU admission. Since the dataset does not have annotated concepts, we calculate the six Sequential Organ Failure Assessment (SOFA) scores (Lambden et al., 2019) and categorise them into 3 levels of severity, for a total of 18 concepts. We use a 3-layer MLP as the concept predictor. Details about the concept selection and hyper-parameters are given in Appendix A.6 and A.7.

5.2 Overall Performance Across Baselines

Our tree-based label predictor produces more trustworthy explanations compared to existing solutions when the concept sets are incomplete. In their work, Barbiero et al. (2022) show that the *Explanation Accuracy* of the Entropy-Net (measured as the average F1 score across all label classes) matches the *Task Accuracy* in most experiments with almost perfect Fidelity scores. However, we observe in this work that this is not guaranteed for datasets with missing concept information. In Table 1, we observe that the fidelity scores of Entropy-Net are lower that 80%, since the logic formulae are missing concept rules and can thus associate some samples to multiple classes. The Explanation Accuracy drops for the same reason. On the other hand, Decision Trees are inherently rule-based models, where no such fidelity issues arise. While their overall Task Accuracy is lower according to Table 1, they exhibit higher Explanation Accuracy with perfect Fidelity. Our Mixed CBM methods allow for leakage inspection and control, to further assist the reliability of explanations.

Mixed CBMs achieve higher task accuracy than their respective hard CBMs and less leakage than their soft counterparts. This can be observed from the results of Table 2 and is expected because the MCBM-Seq and MCBM-Joint models are specialised versions of the same hard/independent CBM, with leaky information concentrated in the subtrees. Soft CBMs generally achieve better performance because they use the concept probabilities from the very first split of the root (see Appendix. 6), thus the Decision Tree finds the optimal splits without restrictions in

Table 1: Explanation Metrics for different Sequential CBMs. Tree-based label predictors achieve higher Explanation Accuracy for sets with incomplete concept information, and do not pose fidelity considerations. Our MCBM-Seq also allows for Leakage Inspection, compared to a Decision Tree.

Method	Morpho-MNIST			CUB			MIMIC-III			Leakage
	Task%	Explanation%	Fidelity%	Task%	Explanation%	Fidelity%	Task%	Explanation%	Fidelity%	Inspection
Seq. (Entropy Net)	52.26	25.01	77.81	65.23	42.37	46.87	83.34	68.94	56.11	✗
Seq. (Decision Tree)	48.42	48.42	100	55.59	55.59	100	83.05	83.05	100	✗
MCBM-Seq	49.62	**49.62**	**100**	47.39	**47.39**	**100**	82.05	**82.05**	**100**	✓

Table 2: Task and Concept Accuracy across different datasets and CBM training methods. Our Mixed CBM methods are comparable with current approaches in overall performance.

	Method	Morpho-MNIST		CUB		MIMIC-III	
		Task%	Concept%	Task%	Concept%	Task%	Concept%
Decision Tree	Hard	47.20	89.94	46.51	94.82	81.65	98.67
	Independent	47.20	89.94	46.51	94.82	81.65	98.67
	MCBM-Seq	49.62	89.94	47.39	94.82	82.05	98.67
	Sequential	48.42	89.94	55.59	94.82	83.05	98.67
	MCBM-Joint ($\lambda_C = 0.1$)	83.16	83.38	56.55	94.21	82.55	97.15
	MCBM-Joint ($\lambda_C = 1$)	67.32	87.92	56.42	94.76	82.05	97.72
	MCBM-Joint ($\lambda_C = 100$)	50.72	90.58	56.29	94.99	81.85	97.82
Entropy Net	Hard	46.31	89.94	56.07	94.82	82.65	98.67
	Independent	45.10	89.94	53.74	94.82	82.55	98.67
	Sequential	52.26	89.94	65.23	94.82	83.34	98.67
	Joint ($\lambda_C = 0.1$)	98.17	83.38	72.26	94.21	83.74	97.15
	Joint ($\lambda_C = 1$)	95.02	87.92	69.37	94.76	83.44	97.72
	Joint ($\lambda_C = 100$)	56.07	90.58	64.12	94.99	82.85	97.82
	Black-Box	99.99	-	84.35	-	92.74	-

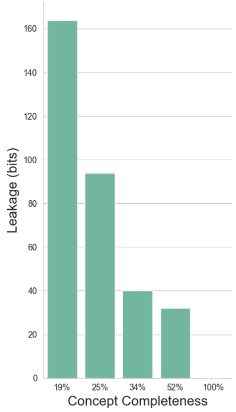


Figure 4: Leakage with Concept Completeness (CUB).

the architecture. However, this leads to Information Leakage in the whole tree and has undesirable effects for interpretability across all decision paths Unlike these, mixed CBMs constrain leakage only to specific leaf nodes, mitigating these undesirable effects throughout the rest of the tree.

Information Leakage decreases when concepts completeness increases. CBMs are more prone to Information Leakage as the number of missing concepts increases. While this issue was also raised by Havasi et al. (2022b), we are able to provide quantitative evidence by measuring the total Information Gain summed across all leaky splits of our Tree. Since the CUB dataset Wah et al. (2011) is concept-complete for class-specific explanations, we evaluate our MCBM-Seq method in Fig. 4 across different levels of completeness, by successively picking more concept groups from the dataset. In the absence of complete concepts, the global tree is small, with more subtrees containing leakage hence enabling more of the leaf nodes to be extended to improve the Information Gain.

5.3 Performance per Decision Path

Our method enables inspecting Information Leakage per decision path. A unique advantage of our method compared to existing CBM training methods is that we can extract performance metrics, inspect and control leakage per individual Decision Paths corresponding to particular groups of data. In Fig. 5, we decompose the full tree from the reduced Morpho-MNIST example in Fig.. 3 into fifteen decision paths, measuring the *Task Accuracy* against the original Hard CBM (global tree) in Table 3. For the three extended decision paths, the accuracy increased from $44\% \rightarrow 45\%$, $37\% \rightarrow 41\%$ and $44\% \rightarrow 57\%$ respectively due to leakage. We also report the Information Gain (Leakage) for each individual leaky split of a sub-tree, if found. We observe that the concept ”length:large” in the leaky split of path 14 in Fig. 5 provides the largest Information Gain (0.230 bits) out of all leaky splits, suggesting that this concept could benefit from additional information. Repeating this for MCBM-Joint, we observe not all the same decision paths are necessarily extended with leakage. The method yields higher task accuracies but also higher information gain (in terms of leakage) Eg.In path 14, the respective leaky split of MCBM-Joint yields a very high accuracy of 80.65%, but the Information Gain (Leakage) from such split is also higher compared to MCBM-Seq (0.968 bits).

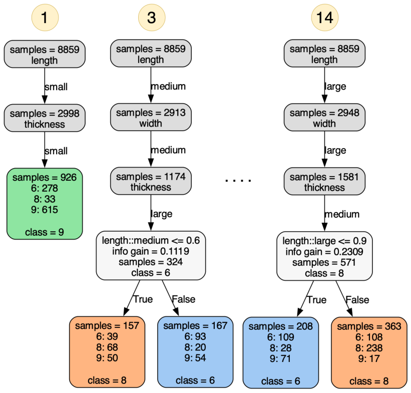


Figure 5: Decomposing the resulting trees of MCBM-Seq into Decision Paths for the Reduced Morpho-MNIST example of Fig. 3.

Table 3: Analysis per path on the Reduced Morpho-MNIST of Fig. 3. Acc. refers to Task Accuracy and IG refers to Information Leakage for each split of the ”leaky” extension.

Path	**Hard**	**MCBM-Seq**		**MCBM-Joint**	
	Acc.%	Acc.%	IG	Acc.%	IG
1	62.39	62.39	-	62.39	-
2	58.53	58.53	-	58.53	-
3	**44.82**	**45.76**	[0.054, 0.006]	**60.45**	[0.191, 0.191]
4	70.48	70.48	-	70.48	-
5	53.92	53.92	-	53.92	-
6	44.20	44.20	-	44.20	-
7	**37.39**	**41.73**	[0.111]	37.39	-
8	55.17	55.17	-	55.17	-
9	52.57	52.57	-	**90.72**	[0.795]
10	58.29	58.29	-	**91.49**	[0.715]
11	99.29	52.57	-	99.29	-
12	92.00	92.00	-	92.00	-
13	39.18	39.18	-	**81.98**	[0.931]
14	**44.91**	**57.70**	[0.230]	**80.65**	[0.968]
15	55.14	55.14	-	**84.19**	[0.911]

Our tree-structure allows for meaningful group-specific explanations. While instance-specific (Entropy-Net) Barbiero et al. (2022) and class-specific explanations can be too complex or generic, group-specific explanations are often more useful, especially in fields like healthcare. Our method allows intuitive control of group size via the minimum samples per leaf (msl) constraint (Appendix A.8). An example of a group-specific explanation using the MCBM-Seq method on the CUB dataset(Wah et al., 2011) is shown in Fig. 2. We consider two bird classes: Red Bellied Woodpecker and Red Headed Woodpecker. At test time, the method traverses the decision path in the global tree and identifies the two classes as indistinguishable based on available concepts. After training a Mixed-Sequential CBM, the label predictor separates the classes using the concept predictor's calibrated probability for “has-breast-pattern-solid.” The predictor finds that Red Bellied Woodpeckers are less than 70% likely to have a solid breast pattern, while Red Headed Woodpeckers are more likely to possess this trait. The full decision path and case-study description are given in Appendix A.10. The user has three options: a) rely solely on the global tree for the most reliable prediction using majority voting, b) extend the decision process with MCBM-Seq for more intuitive and higher-performance results, or c) use MCBM-Joint's less intuitive probabilities for maximum accuracy. Importantly, unlike a purely soft CBM, leakage will not impact all decision-making paths in a mixed CBM and will be isolated to only some leaf nodes that can be extended.

6 Conclusion

In this work, we introduce MCBM-Seq and MCBM-Joint methods that use decision trees to inspect and control information leakage in CBMs. These tree-based approaches maintain high fidelity in the explanations and achieve better accuracy on datasets with incomplete concept information. Unlike purely Soft CBMs, the mixed concept representations limit information leakage in data subsets with insufficient concept information. They also quantify leakage per decision path and rule, producing more meaningful group-based explanations.

A limitation of our work is that the final, merged trees can be difficult to visualise and inspect for very large datasets. Yet, we show that a leakage analysis for specific decision paths of interest can also be meaningful. Also, *mixed* CBMs allow us to derive more interpretable and reliable explanations but typically do not match the model accuracy of purely Soft CBMs. A promising avenue for future work is to further address the problem of missing concept information by combining our approach with concept discovery strategies. While information leakage makes up for some of the missing concept information, these discovery strategies could be optimised per decision path in our tree-structure, to further distinguish groups of samples that cannot take advantage of leakage. Moreover, our method does not pose any constraint on the architecture of the concept encoder, thus it could also integrate a more expressive Auto-Regressive Concept encoder or a Stochastic Concept Encoder to further assist the concept predictions.

REFERENCES

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Pietro Lió, Marco Gori, and Stefano Melacci. Entropy-based logic explanations of neural networks. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(6):6046–6054, Jun. 2022. doi: 10.1609/aaai.v36i6.20551. URL https://ojs.aaai.org/index.php/AAAI/article/view/20551.

Leo Breiman, Jerome Friedman, Charles J. Stone, and R.A. Olshen. *Classification and Regression Trees*. Chapman and Hall/CRC, 1984.

Daniel C Castro, Jeremy Tan, Bernhard Kainz, Ender Konukoglu, and Ben Glocker. Morpho-MNIST: Quantitative assessment and diagnostics for representation learning. (178):1–29, September 2018.

Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, 2020.

Gabriele Ciravegna, Francesco Giannini, Marco Gori, Marco Maggini, and Stefano Melacci. Human-driven fol explanations of deep learning. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence*, IJCAI'20, 2021. ISBN 9780999241165.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. *Advances in neural information processing systems*, 32, 2019.

Shantanu Ghosh, Ke Yu, Forough Arabshahi, and Kayhan Batmanghelich. Dividing and conquering a BlackBox to a mixture of interpretable models: Route, interpret, repeat. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11360–11397. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/ghosh23c.html.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1321–1330, 2017.

Trevor Hastie and Robert Tibshirani. Generalized additive models; some applications. In Robert Gilchrist, Brian Francis, and Joe Whittaker (eds.), *Generalized Linear Models*, pp. 66–81, New York, NY, 1985. Springer US. ISBN 978-1-4615-7070-7.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 23386–23397. Curran Associates, Inc., 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/944ecf65a46feb578a43abfd5cddd960-Paper-Conference.pdf.

Marton Havasi, Sonali Parbhoo, and Finale Doshi-Velez. Addressing leakage in concept bottleneck models. *Advances in Neural Information Processing Systems*, 35:23386–23397, 2022b.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. In *Proceedings of 2016 IEEE Conference on Computer Vision and Pattern Recognition*, CVPR '16, pp. 770–778. IEEE, June 2016. doi: 10.1109/CVPR.2016.90. URL http://ieeexplore.ieee.org/document/7780459.

Lena Heidemann, Maureen Monnet, and Karsten Roscher. Concept correlation and its effects on concept-based models. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pp. 4780–4788, January 2023.

Xiyang Hu, Cynthia Rudin, and Margo Seltzer. Optimal sparse decision trees. *Advances in Neural Information Processing Systems*, 32, 2019.

Alistair E. W. Johnson, Lucas Bulgarelli, Lu Shen, Alvin Gayles, Ayad Shammout, Steven Horng, Tom J. Pollard, Benjamin Moody, Brian Gow, Li wei H. Lehman, Leo Anthony Celi, and Roger G. Mark. Mimic-iv, a freely accessible electronic health record dataset. *Scientific Data*, 10, 2023. URL https://api.semanticscholar.org/CorpusID:255439889.

Gordon V. Kass. An exploratory technique for investigating large quantities of categorical data. *Journal of The Royal Statistical Society Series C-applied Statistics*, 29:119–127, 1980. URL https://api.semanticscholar.org/CorpusID:61329067.

Eunji Kim, Dahuin Jung, Sangha Park, Siwon Kim, and Sungroh Yoon. Probabilistic concept bottleneck models. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Pang Wei Koh, Thao Nguyen, Yew Siang Tang, Stephen Mussmann, Emma Pierson, Been Kim, and Percy Liang. Concept bottleneck models. In Hal Daumé Iii and Aarti Singh (eds.), *Proceedings of the 37th International Conference on Machine Learning*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5338–5348. PMLR, 2020.

Neeraj Kumar, Alexander C Berg, Peter N Belhumeur, and Shree K Nayar. Attribute and simile classifiers for face verification. In *2009 IEEE 12th international conference on computer vision*, pp. 365–372. IEEE, 2009.

Simon Lambden, Pierre-François Laterre, Mitchell M. Levy, and Bruno François. The sofa score—development, utility and challenges of accurate assessment in clinical trials. *Critical Care*, 23, 2019. URL https://api.semanticscholar.org/CorpusID:208302509.

Christoph H Lampert, Hannes Nickisch, and Stefan Harmeling. Learning to detect unseen object classes by between-class attribute transfer. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 951–958. IEEE, 2009.

Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. doi: 10.1109/5.726791.

Klas Leino, Shayak Sen, Anupam Datta, Matt Fredrikson, and Linyi Li. Influence-directed explanations for deep convolutional networks. In *2018 IEEE international test conference (ITC)*, pp. 1–8. IEEE, 2018.

Anita Mahinpei, Justin Clark, Isaac Lage, Finale Doshi-Velez, and Weiwei Pan. Promises and pitfalls of black-box concept learning models. *CoRR*, abs/2106.13314, 2021. URL https://arxiv.org/abs/2106.13314.

Emanuele Marconato, Andrea Passerini, and Stefano Teso. Glancenets: Interpretable, leak-proof concept-based models. *Advances in Neural Information Processing Systems*, 35:21212–21227, 2022.

Andrei Margeloiu, Matthew Ashman, Umang Bhatt, Yanzhi Chen, Mateja Jamnik, and Adrian Weller. Do concept bottleneck models learn as intended? 2021. doi: 10.17863/CAM.80941. URL https://www.repository.cam.ac.uk/handle/1810/333521.

Richard D. McKelvey and William James Zavoina. A statistical model for the analysis of ordinal level dependent variables. *Journal of Mathematical Sociology*, 4:103–120, 1975. URL https://api.semanticscholar.org/CorpusID:121505375.

Tuomas Oikarinen, Subhro Das, Lam M. Nguyen, and Tsui-Wei Weng. Label-free concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=FlCg47MNvBA.

JC PLATT. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in Large Margin Classifiers*, 1999.

Cynthia Rudin. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nature machine intelligence*, 1(5):206–215, 2019.

Habiba Muhammad Sani, Ci Lei, and Daniel Neagu. Computational complexity analysis of decision tree algorithms. In Max Bramer and Miltos Petridis (eds.), *Artificial Intelligence XXXV*, pp. 191–197, Cham, 2018. Springer International Publishing. ISBN 978-3-030-04191-5.

Andrei Semenov, Vladimir Ivanov, Aleksandr Beznosikov, and Alexander Gasnikov. Sparse concept bottleneck models: Gumbel tricks in contrastive learning. *arXiv preprint arXiv:2404.03323*, 2024.

Chenming Shang, Shiji Zhou, Hengyuan Zhang, Xinzhe Ni, Yujiu Yang, and Yuwang Wang. Incremental residual concept bottleneck models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 11030–11040, 2024.

Moritz Vandenhirtz, Sonia Laguna, Ričards Marcinkevičs, and Julia E. Vogt. Stochastic concept bottleneck models, 2024. URL https://arxiv.org/abs/2406.19272.

Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. *The Caltech-UCSD Birds-200-2011 Dataset*. Jul 2011.

Mike Wu, Michael Hughes, Sonali Parbhoo, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Beyond sparsity: Tree regularization of deep models for interpretability. *Proc. Conf. AAAI Artif. Intell.*, 32(1), April 2018.

Mike Wu, Sonali Parbhoo, Michael Hughes, Ryan Kindle, Leo Celi, Maurizio Zazzi, Volker Roth, and Finale Doshi-Velez. Regional tree regularization for interpretability in deep neural networks. *Proc. Conf. AAAI Artif. Intell.*, 34(04):6413–6421, April 2020.

Chih-Kuan Yeh, Been Kim, Sercan Arik, Chun-Liang Li, Tomas Pfister, and Pradeep Ravikumar. On completeness-aware concept-based explanations in deep neural networks. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin (eds.), *Advances in Neural Information Processing Systems*, volume 33, pp. 20554–20565. Curran Associates, Inc., 2020. URL https://proceedings.neurips.cc/paper_files/paper/2020/file/ecb287ff763c169694f682af52c1f309-Paper.pdf.

Mert Yuksekgonul, Maggie Wang, and James Zou. Post-hoc concept bottleneck models. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=nA5AZ8CEyow.

Renos Zabounidis, Ini Oguntola, Konghao Zhao, Joseph Campbell, Simon Stepputtis, and Katia Sycara. Benchmarking and enhancing disentanglement in concept-residual models, 2023. URL https://arxiv.org/abs/2312.00192.

Mateo Espinosa Zarlenga, Pietro Barbiero, Gabriele Ciravegna, Giuseppe Marra, Francesco Giannini, Michelangelo Diligenti, Zohreh Shams, Frederic Precioso, Stefano Melacci, Adrian Weller, Pietro Lio, and Mateja Jamnik. Concept embedding models: beyond the accuracy-explainability trade-off. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA, 2024. Curran Associates Inc. ISBN 9781713871088.

Bolei Zhou, David Bau, Aude Oliva, and Antonio Torralba. Interpreting deep visual representations via network dissection. *IEEE transactions on pattern analysis and machine intelligence*, 41(9): 2131–2145, 2018.

A Appendix

A.1 Reproducibility Statement

To ensure reproducibility, all experiments described in this paper were conducted with fixed random seeds, which are explicitly detailed in the provided code. In addition, all scripts necessary to replicate the main results will be made publicly accessible. Comprehensive instructions for running the experiments and verifying the outcomes will also be included in the repository, alongside the random seed values for each specific experiment. The anonymous code is given here: `https://anonymous.4open.science/r/ICLR_2025_Controlling_IL_With_Trees-75EA/README.md`

A.2 Using Trees to quantify Information Leakage

As in section 4.2, consider again the subset of samples s that end up in one of the leaf nodes of the global tree, and a soft concept $\hat{c}_k$ based on which we perform the first split in the Mixed Sequential CBM. Consider also the two subsets after the split s_1 and s_2, and the target distributions y_s, y_{s_1} and y_{s_2} respectively. According to Breiman et al. (1984), the **Information Gain** of a split in a decision tree can be defined as the difference of the entropy of the subset s before the split and the weighted sum of the entropy's of the two nodes after the split using the concept $\hat{c}_k$:

If c_s represents the set of hard concepts that appear in the decision path of the global tree up to the leaf node of the subset s, then we assume that

$$H(y|c_s) \approx H(y_s) \tag{6}$$

In other words, we have already taken into account the information present in the concepts c_s to construct the target distribution of this path y_s from the initial target distribution of all samples y when building the global tree. Similarly, given we have the information from the hard concepts of the path c_s and the new soft concept $\hat{c}_k$, we assume that:

$$H(y|\hat{c}_k, c_s) \approx \left[\frac{|s_1|}{|s|}H(y_{s_1}) + \frac{|s_2|}{|s|}H(y_{s_2})\right] \tag{7}$$

Thus, we can approximate the Information Leakage given to the label encoder by the soft concept $\hat{c}_k$ according to Eq. 4:

$$\begin{aligned} I_{\text{Leakage}}(\hat{c}_k) &= I(y; \hat{c}_k\, | c_s) = H(y|c_s) - H(y|\hat{c}_k, c_s) && (8) \\ &\overset{\text{equation 6 equation 7}}{\approx} H(y_s) - \left[\frac{|s_1|}{|s|}H(y_{s_1}) + \frac{|s_2|}{|s|}H(y_{s_2})\right] && (9) \\ &\overset{\text{equation 5}}{=} IG(\hat{c}_k) && (10) \end{aligned}$$

Thus, using Decision Trees, we can use the Information Gain we receive when splitting a node with a soft concept $\hat{c}_k$ as a measure of the Information Leakage that the soft concept provided. **This is only valid because of the way we constructed our tree**, by first using the hard concepts to fit the global tree and then specialising the leaf nodes with soft concept splits.

A.3 Leakage Inspection: A naive solution

A naive attempt to inspect any Information Leakage that the Soft Model exploited compared to a Hard CBM is to first train the same concept predictor for both models, and then use a separate Decision Tree classifier for each CBM as label predictor. In the first case, the inputs of the Tree are binary, ground truth concepts, whereas in the second case the inputs are the predicted concept probabilities of the concept encoder. To answer the question, we then need to compare two Trees. For reference, we visualise the Decision Trees trained for the Morpho-MNIST example Castro et al. (2018) with a constraint on the minimum samples per lead of $msl = 150$. The trees are shown side-by-side in Fig. 6.

We immediately observe that the task is not trivial because the structure of the two trees can be very different. Even if the root node uses the same concept in both trees to perform the primary split (e.g. ”length xlarge”), the different threshold value used in each case may cause the resulting subsets to be completely different. Thus, the two trees may be incomparable after the first split, since the respective children nodes may use a completely different structure to distinguish their samples.

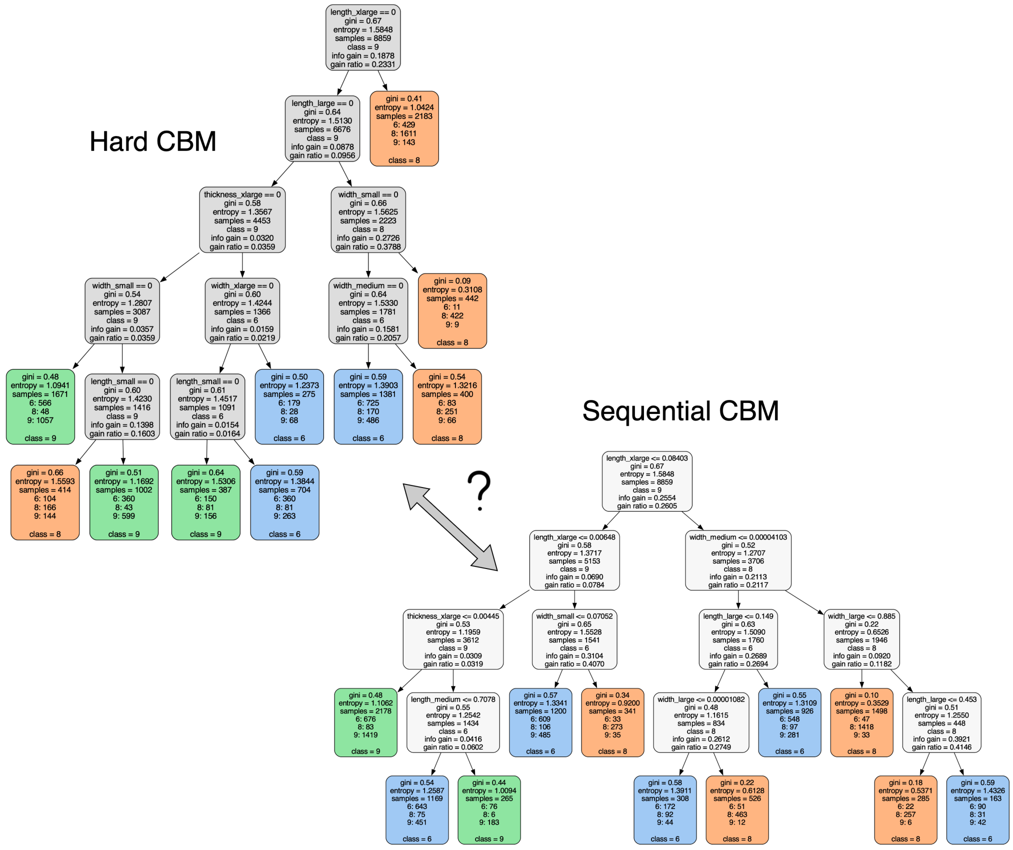


Figure 6: Leakage Inspection: Naive Comparison of a Hard and a Soft Sequential CBM.

A.4 Complexity Analysis for the label predictor of MCBM-Seq

An advantage of our approach is the limited computational overhead compared to training a standard independent CBM, because the concept encoder is trained only once. The individual Sequential CBMs all share the same concept predictor. Thus, the only overhead is that of training a sub-tree for each leaf of the global tree. In practice, this added computational cost is minimal because each tree only has access to a small subset n_i of the total samples n, where $\sum_i^d n_i = n$ and d is the number of leaf nodes in the global tree.

More specifically, the time complexity of a Decision Tree is $O(mn\,log_2 n)$ according to Sani et al. (2018), where n is the total number of samples and m is the number of attributes. This holds because the computational cost of performing a split based on one attribute follows the recursion formula of the divide and conquer algorithm $O(n\,log_2 n)$, and this process is repeated for all m attributes to find the best split. Thus, the total cost J of training the label predictor in the Leakage Inspection algorithm can be written as the cost of training the global tree and all its sub-trees:

$$J = J_{global} + \sum_i^d J_i \longrightarrow O(mn\,log_2 n) \tag{11}$$

The complexity is thus equivalent to that of a purely soft Sequential CBM that uses a single decision tree as label predictor, since the msl constraint of the global and sub-trees is the same.

A.5 MCBM-JOINT

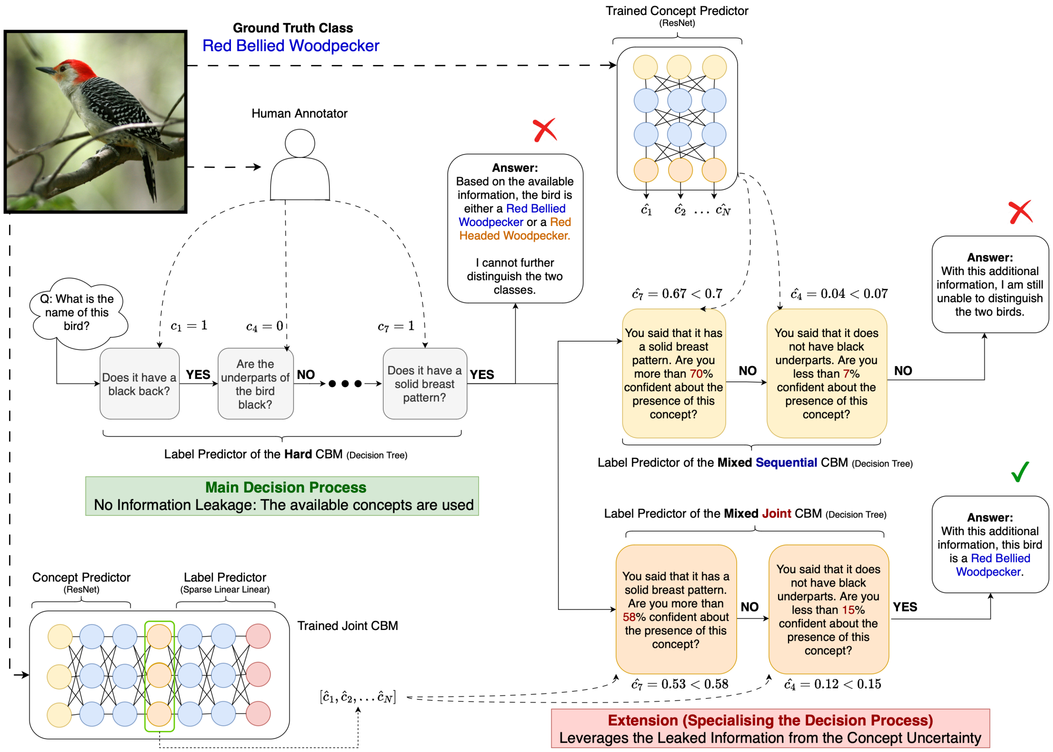

Ground Truth Class
Red Bellied Woodpecker
Human Annotator
Trained Concept Predictor
(ResNet)
$\hat{c}_1$ $\hat{c}_2$ … $\hat{c}_N$
Answer:
Based on the available information, the bird is either a Red Bellied Woodpecker or a Red Headed Woodpecker.
I cannot further distinguish the two classes.
Q: What is the name of this bird?
$c_1 = 1$
$c_4 = 0$
$c_7 = 1$
Does it have a black back?
YES
Are the underparts of the bird black?
NO
Does it have a solid breast pattern?
YES
Label Predictor of the Hard CBM (Decision Tree)
Main Decision Process
No Information Leakage: The available concepts are used
$\hat{c}_7 = 0.67 < 0.7$
$\hat{c}_4 = 0.04 < 0.07$
You said that it has a solid breast pattern. Are you more than 70% confident about the presence of this concept?
NO
You said that it does not have black underparts. Are you less than 7% confident about the presence of this concept?
NO
Answer:
With this additional information, I am still unable to distinguish the two birds.
Label Predictor of the Mixed Sequential CBM (Decision Tree)
Label Predictor of the Mixed Joint CBM (Decision Tree)
You said that it has a solid breast pattern. Are you more than 58% confident about the presence of this concept?
NO
You said that it does not have black underparts. Are you less than 15% confident about the presence of this concept?
YES
Answer:
With this additional information, this bird is a Red Bellied Woodpecker.
$\hat{c}_7 = 0.53 < 0.58$
$\hat{c}_4 = 0.12 < 0.15$
Concept Predictor
(ResNet)
Label Predictor
(Sparse Linear Linear)
Trained Joint CBM
$[\hat{c}_1, \hat{c}_2, \dots \hat{c}_N]$
Extension (Specialising the Decision Process)
Leverages the Leaked Information from the Concept Uncertainty

Figure 7: The Decision Making Process of The Mixed Joint CBM Algorithm (MCBM-Joint) when classifying an image with annotated concepts. The concept probabilities of the Joint CBM can be used to specialise the decision-making process of the label predictor when a) the available annotated concepts are not sufficient, and b) the concept probabilities of the Sequential CBM are also not sufficient (The probability numbers shown are constructed only for visualisation purposes).

The **Mixed Joint CBM algorithm (MCBM-Joint)** is a direct extension to MCBM-Seq and the underlying logic is presented in Fig. 7. For the given example with annotated concepts, both the Decision Tree of the Hard CBM and its path extension of MCBM-Seq cannot identify the class of the bird. If we also have in our disposal a trained Joint CBM, we could repeat the algorithm and find a new path extension using the concept probabilities from the concept predictor of the Joint CBM. This time, the updated decision path is able to make the correct distinction.

A.6 DATASETS: ADDITIONAL DETAILS

CUB: The dataset is concept-complete, as all classes can be distinguished based on the ground truth annotated concepts at training time. To simulate the scenario of missing concept information, we select 45 out of the 112 concepts by picking all concepts from the following concept groups: ”has-bill-shape”, ”has-wing-color”, ”has-upperparts-color”, ”has-underparts-color”, ”has-breast-pattern”, ”has-back-color”, ”has-upper-tail-color”, ”has-breast-color”.

MIMIC-IV: From the total number of ICU admissions, we extract those patients with only one reported ICU Admission and whose age is between 18 and 90 years old. Moreover, for our mortality prediction task, we only select patients that were in the ICU for 48 hours or less. We use the following measurements as features for the length of their stay: creatinine, urine, norepinephrine, epinephrine, dobutamine, dopamine, mean blood pressure, P/F ratio, num. plalelets. Using these features, we calculate the SOFA scores Lambden et al. (2019), which provide an assessment for the condition of the following systems in the human body: respiratory, coagulation, liver (hepatic), cardiovascular, neurological (Glasgow coma scale) and renal. To construct concepts, we annotate

Algorithm 2 Mixed Joint CBM Training (MCBM-Joint)

```
Input: $N = \{1, ..., n\}$ samples; $K = \{1, ..., k\}$ concepts; $R = \{1, ..., r\}$ classes;
       A dataset $D_{train} = \{X, C, Y\}$, where $X \subset \mathbb{R}^d$, $C \subset \{0,1\}^k$, $Y \subset \{0,1\}^r$;
       A set of $N_{test}$ samples with $X_{test} \subset \mathbb{R}^d$; Minimum Samples per leaf ($msl$).
Output: A hard tree: $T_{hard}$; a set of soft trees using concept probabilities from a
        Sequential CBM: $T_{seq} = \{T_1, ..., T_M\}$; a set of soft trees using concept
        probabilities from a Joint CBM: $T_{joint} = \{T_1, ..., T_M\}$; test predictions $\hat{Y}$.
procedure TRAINING($D_{train}, msl$)
    Train the concept predictor $f : X \rightarrow \hat{C}$, where $\hat{C} \subset [0,1]^k$ (Eq. 1);
    Fine-Tune the predictor $f_{joint}$ using the Joint-Training objective of Eq .3;
    ...
    return $T_{hard}, T_{seq}, T_{joint}$
end procedure

procedure TRAINING SUB-TREE($D_{train}, f, T_{hard}$)
    (Same as in Algorithm 1);
    return $T_{soft}$
end procedure

procedure EVALUATION($X_{test}, f, T_{hard}, T_{joint}$)
    (Same as in Algorithm 1);
    return $\hat{Y}$
end procedure
```

SOFA scores 0-1 as concept 0 (normal), 2-3 as concept 1 (moderate), and 4 as concept 2 (severe). We thus have 18 concepts, 3 per system.

A.7 HYPER-PARAMETER SETTING

We provide the hyper-parameters used for training the independent Concept and Label Predictors of MCBM-Seq. In the case of MCBM-Joint, an additional fine-tuning step of the concept predictor is performed using the Joint objective of Eq. 3. We test either the CHAID (Kass, 1980) or the CART (Breiman et al., 1984) algorithm to train the Decision Trees.

Table 4: Hyperparameter setting of MCBM-Seq, MCBM-Joint used by Morpho-MNIST, CUB-200 and MIMIC-IV

Setting	**Morpho-MNIST**	**CUB-200**	**MIMIC-IV**
Concept Predictor	LeNet	ResNet-18	MLP: [128,64,2]
Learning rate ($x \rightarrow c$)	0.001	0.001	0.01
Optimiser	Adam	Adam	Adam
Weight-decay	0.00001	0.00001	0
Epochs	200	300	50
Decision Tree Algorithm used	CHAID	CART	CHAID
Minimum samples per leaf (msl)	1	150	30
Probability Calibration	Temperature	Platt	Temperature
Joint-training epochs	50	100	50

A.8 Controlling the group size

A.8.1 Morpho-MNIST

We test the size of the resulting trees compared to the performance accuracy for the Morpho-MNIST dataset. We observe that the $\mathbf{msl}$ constraint controls the Performance/Interpretability trade-off. As expected, both the number of Tree nodes and the Task Accuracy increase as the msl constraint is reduced. However, an msl close to 1 may not improve performance or even result in over-fitting. The advantage of the $\mathbf{msl}$ hyper-parameter is that it is human intuitive, as it directly controls the minimum size of our desired groups to perform explanations.

$\mathbf{msl} = \mathbf{150}$	**MCBM-Seq**	**MCBM-Joint**
		$\lambda_C = 100$
Task Accuracy	0.496	0.560
Concept Accuracy	0.894	0.905
Num. Tree Nodes	147	155

$\mathbf{msl} = \mathbf{20}$	**MCBM-Seq**	**MCBM-Joint**
		$\lambda_C = 100$
Task Accuracy	0.551	0.595
Concept Accuracy	0.894	0.905
Num. Tree Nodes	795	756

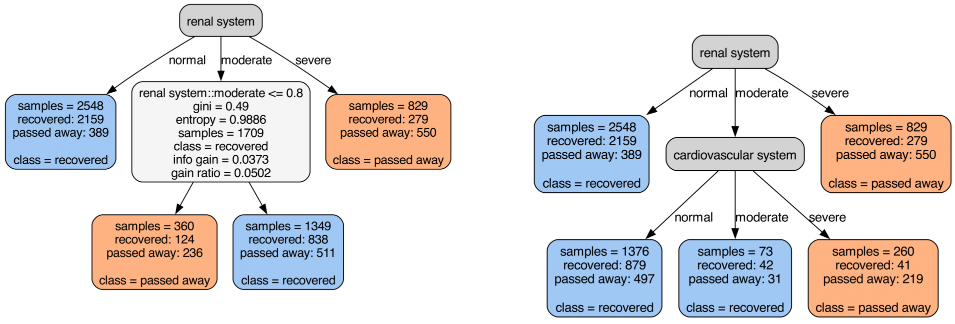
$\mathbf{msl} = \mathbf{5}$	**MCBM-Seq**	**MCBM-Joint**
		$\lambda_C = 100$
Task Accuracy	0.552	0.708
Concept Accuracy	0.894	0.905
Num. Tree Nodes	2455	2406

Table 5: Performance of MCBM-Seq, MCBM-Joint on Morpho-MNIST, with decreasing msl.

A.8.2 Mimic-IV

In this section we demonstrate that the msl constraint does not only affect the Task Accuracy and the total size of the MCBM-Seq or MCBM-Joint tree, but also the meaning of our explanations in terms of the Information Leakage inspected by either method.

In the following figures, the MIMIC-IV trees are shown after the application of the MCBM-Seq algorithm for low, medium and high msl values. The nodes annotated dark grey show the splits of the global tree, and the nodes with light grey show those of a leaky extension, if found. When $msl = 150$, we observe that the tree is too restricted for this small-sized dataset. Yet, the MCBM-Seq is able to find a leaky split under this constraint for patients whose renal system is in moderate condition (SOFA score 2-3). If the concept predictor is more than 80% confident that the condition of a given patient is indeed moderate, the label predictor is more confident that this patient may live. When the constraint is slightly relaxed at $msl = 70$, we observe that the corresponding tree can split the same group of patients with a second hard concept (condition of the cardiovascular system) and does not need to use Information Leakage. Thus, this constraint fits better for this dataset. For medium-sized groups, using an $msl = 50$, we observe that the predictor can fit the data even better, without the presence of leakage. However, when $msl = 30$, the two discovered leaky rules are very non-intuitive: When the concept predictor is more than 90% confident that the renal system is in severe condition, it is more likely that a patient may live. When a rule is very non-intuitive, it indicates a high noise in concept annotations. On the contrary, the leaky rule described in the first tree aligns with human intuition. This case-study shows that tuning the msl constraint is critical for the performance and interpretability of our methods. Yet, this is the only single hyper-parameter that our label-predictor has.



(a) MCBM-Seq on MIMIC-IV: $msl = 150$.

(b) MCBM-Seq on MIMIC-IV: $msl = 70$.

Figure 8: MCBM-Seq on MIMIC-IV with groups of large size.

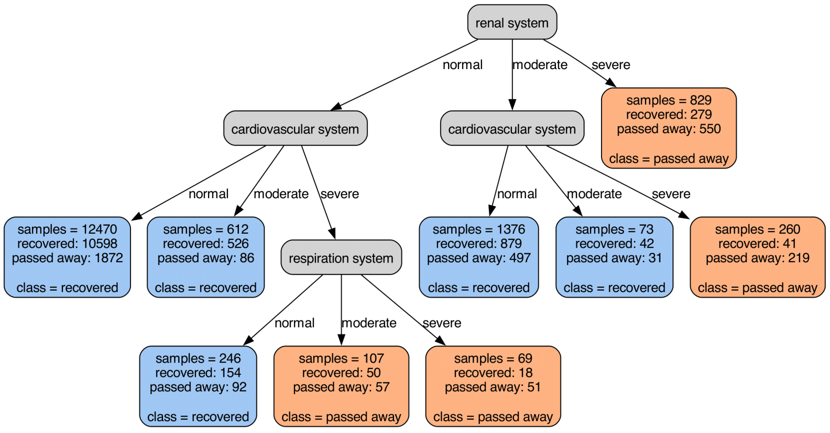


Figure 9: MCBM-Seq on MIMIC-IV with groups of medium size: $msl = 50$.

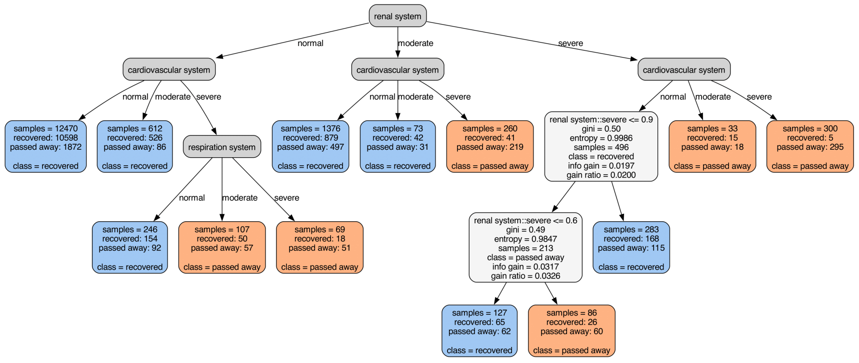


Figure 10: MCBM-Seq on MIMIC-IV with groups of small size: $msl = 30$.

A.9 MCBM-Seq on Morpho-MNIST: Full Trees

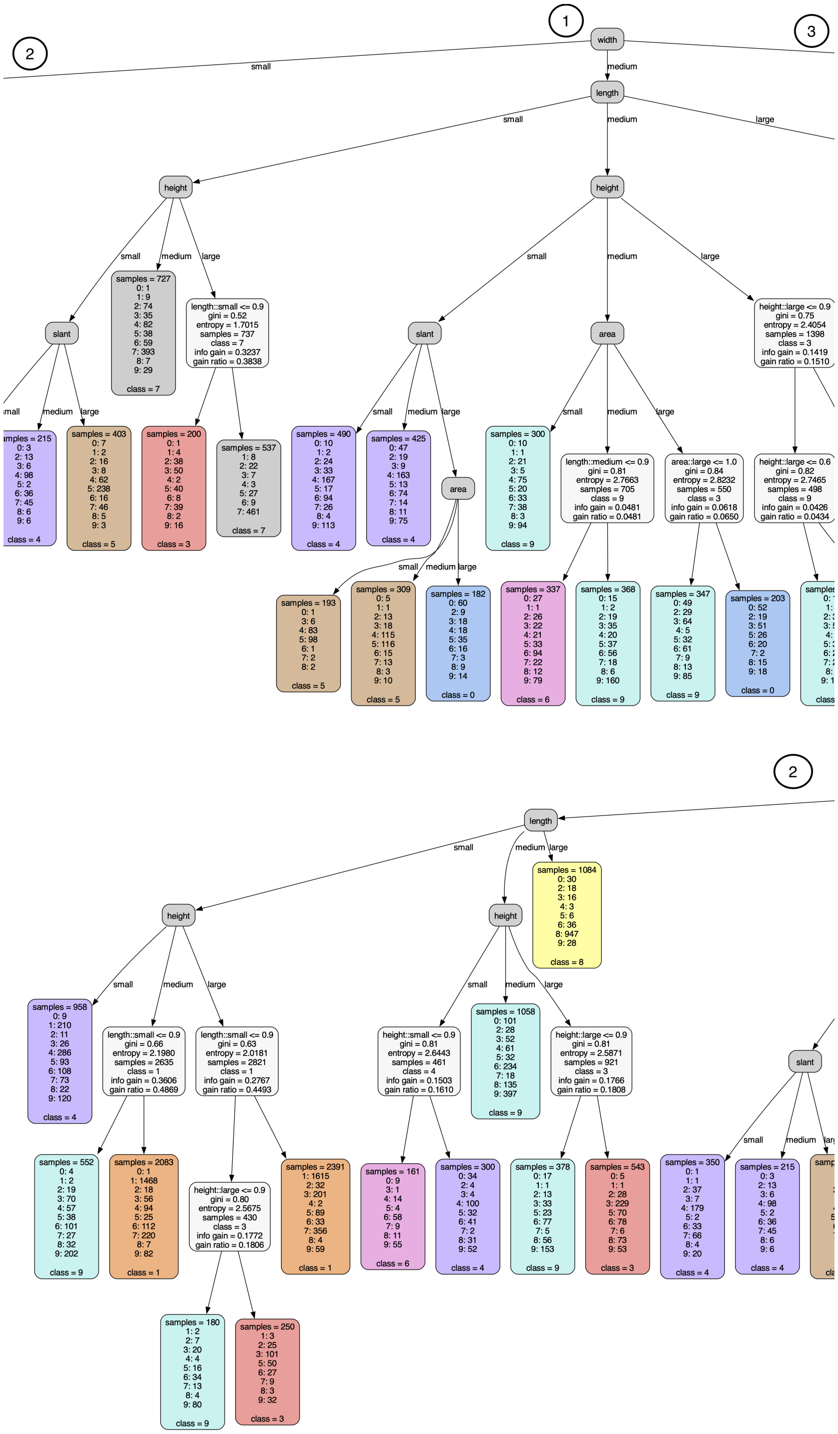

1
2
3
width
small
medium
length
small
medium
large
height
small
medium
large
slant
small
medium
large
samples = 215 0: 3 2: 13 3: 6 4: 98 5: 2 6: 36 7: 45 8: 6 9: 6 class = 4
samples = 403 0: 7 1: 2 2: 16 3: 8 4: 62 5: 238 6: 16 7: 46 8: 5 9: 3 class = 5
samples = 727 0: 1 1: 9 2: 74 3: 35 4: 82 5: 38 6: 59 7: 393 8: 7 9: 29 class = 7
length::small <= 0.9 gini = 0.52 entropy = 1.7015 samples = 737 class = 7 info gain = 0.3237 gain ratio = 0.3838
samples = 200 0: 1 1: 4 2: 38 3: 50 4: 2 5: 40 6: 8 7: 39 8: 2 9: 16 class = 3
samples = 537 1: 8 2: 22 3: 7 4: 3 5: 27 6: 9 7: 461 class = 7
height
small
medium
large
slant
small
medium
large
samples = 490 0: 10 1: 2 2: 24 3: 33 4: 167 5: 17 6: 94 7: 26 8: 4 9: 113 class = 4
samples = 425 0: 47 2: 19 3: 9 4: 163 5: 13 6: 74 7: 14 8: 11 9: 75 class = 4
area
small
medium
large
samples = 193 0: 1 3: 6 4: 83 5: 98 6: 1 7: 2 8: 2 class = 5
samples = 309 0: 5 1: 1 2: 13 3: 18 4: 115 5: 116 6: 15 7: 13 8: 3 9: 10 class = 5
samples = 182 0: 60 2: 9 3: 18 4: 18 5: 35 6: 16 7: 3 8: 9 9: 14 class = 0
area
small
medium
large
samples = 300 0: 10 1: 1 2: 21 3: 5 4: 75 5: 20 6: 33 7: 38 8: 3 9: 94 class = 9
length::medium <= 0.9 gini = 0.81 entropy = 2.7663 samples = 705 class = 9 info gain = 0.0481 gain ratio = 0.0481
samples = 337 0: 27 1: 1 2: 26 3: 22 4: 21 5: 33 6: 94 7: 22 8: 12 9: 79 class = 6
samples = 368 0: 15 1: 2 2: 19 3: 35 4: 20 5: 37 6: 56 7: 18 8: 6 9: 160 class = 9
area::large <= 1.0 gini = 0.84 entropy = 2.8232 samples = 550 class = 3 info gain = 0.0618 gain ratio = 0.0650
samples = 347 0: 49 2: 29 3: 64 4: 5 5: 32 6: 61 7: 9 8: 13 9: 85 class = 9
samples = 203 0: 52 2: 19 3: 51 5: 26 6: 20 7: 2 8: 15 9: 18 class = 0
height::large <= 0.9 gini = 0.75 entropy = 2.4054 samples = 1398 class = 3 info gain = 0.1419 gain ratio = 0.1510
height::large <= 0.6 gini = 0.82 entropy = 2.7465 samples = 498 class = 9 info gain = 0.0426 gain ratio = 0.0434
2
length
small
medium
large
samples = 1084 0: 30 2: 18 3: 16 4: 3 5: 6 6: 36 8: 947 9: 28 class = 8
height
small
medium
large
samples = 958 0: 9 1: 210 2: 11 3: 26 4: 286 5: 93 6: 108 7: 73 8: 22 9: 120 class = 4
length::small <= 0.9 gini = 0.66 entropy = 2.1980 samples = 2635 class = 1 info gain = 0.3606 gain ratio = 0.4869
samples = 552 0: 4 1: 2 2: 19 3: 70 4: 57 5: 38 6: 101 7: 27 8: 32 9: 202 class = 9
samples = 2083 0: 1 1: 1468 2: 18 3: 56 4: 94 5: 25 6: 112 7: 220 8: 7 9: 82 class = 1
length::small <= 0.9 gini = 0.63 entropy = 2.0181 samples = 2821 class = 1 info gain = 0.2767 gain ratio = 0.4493
height::large <= 0.9 gini = 0.80 entropy = 2.5675 samples = 430 class = 3 info gain = 0.1772 gain ratio = 0.1806
samples = 180 1: 2 2: 7 3: 20 4: 4 5: 16 6: 34 7: 13 8: 4 9: 80 class = 9
samples = 250 1: 3 2: 25 3: 101 5: 50 6: 27 7: 9 8: 3 9: 32 class = 3
samples = 2391 1: 1615 2: 32 3: 201 4: 2 5: 89 6: 33 7: 356 8: 4 9: 59 class = 1
height
small
medium
large
height::small <= 0.9 gini = 0.81 entropy = 2.6443 samples = 461 class = 4 info gain = 0.1503 gain ratio = 0.1610
samples = 161 0: 9 3: 1 4: 14 5: 4 6: 58 7: 9 8: 11 9: 55 class = 6
samples = 300 0: 34 2: 4 3: 4 4: 100 5: 32 6: 41 7: 2 8: 31 9: 52 class = 4
samples = 1058 0: 101 2: 28 3: 52 4: 61 5: 32 6: 234 7: 18 8: 135 9: 397 class = 9
height::large <= 0.9 gini = 0.81 entropy = 2.5871 samples = 921 class = 3 info gain = 0.1766 gain ratio = 0.1808
samples = 378 0: 17 1: 1 2: 13 3: 33 5: 23 6: 77 7: 5 8: 56 9: 153 class = 9
samples = 543 0: 5 1: 1 2: 28 3: 229 5: 70 6: 78 7: 6 8: 73 9: 53 class = 3
slant
small
medium
samples = 350 0: 1 1: 1 2: 37 3: 7 4: 179 5: 2 6: 33 7: 66 8: 4 9: 20 class = 4
samples = 215 0: 3 2: 13 3: 6 4: 98 5: 2 6: 36 7: 45 8: 6 9: 6 class = 4

Figure 11: **PART 1**: MCBM-Seq - Full Merged Tree on Morpho-MNIST with $msl = 150$, split in four images. The dark grey nodes show the decision rules of the global tree, and do not introduce leakage. The white grey nodes show the leaky splits of the corresponding sub-tree, if found. Indicators in the top of the pictures indicate where the following or previous picture starts. Indicator 1 shows the root of the tree.

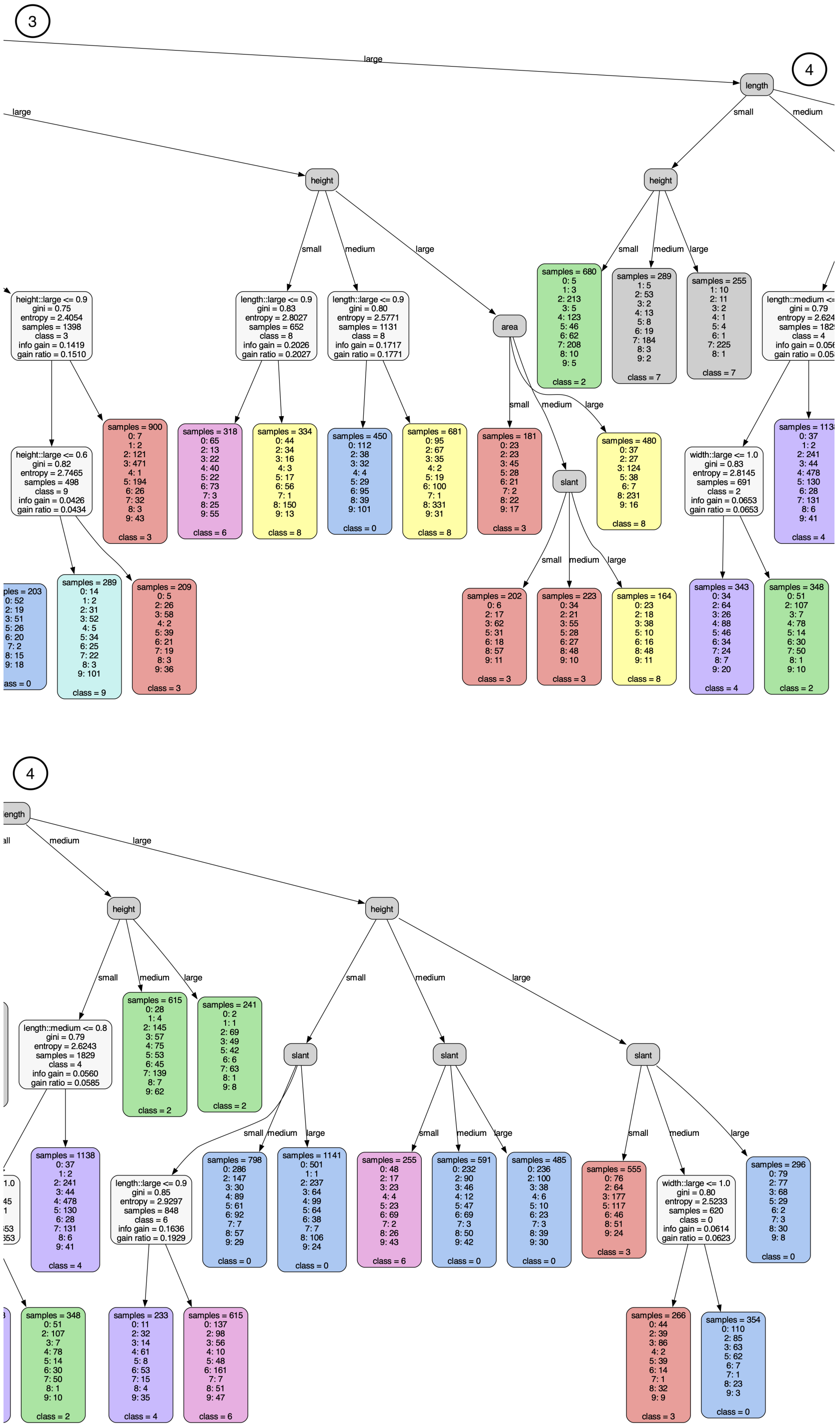

3
large
4
length
small
medium
large
height
small
medium
large
height
small
medium
large
height::large <= 0.9 gini = 0.75 entropy = 2.4054 samples = 1398 class = 3 info gain = 0.1419 gain ratio = 0.1510
length::large <= 0.9 gini = 0.83 entropy = 2.8027 samples = 652 class = 8 info gain = 0.2026 gain ratio = 0.2027
length::large <= 0.9 gini = 0.80 entropy = 2.5771 samples = 1131 class = 8 info gain = 0.1717 gain ratio = 0.1771
area
samples = 680 0: 5 1: 3 2: 213 3: 5 4: 123 5: 46 6: 62 7: 208 8: 10 9: 5 class = 2
samples = 289 1: 5 2: 53 3: 2 4: 13 5: 8 6: 19 7: 184 8: 3 9: 2 class = 7
samples = 255 1: 10 2: 11 3: 2 4: 1 5: 4 6: 1 7: 225 8: 1 class = 7
height::large <= 0.6 gini = 0.82 entropy = 2.7465 samples = 498 class = 9 info gain = 0.0426 gain ratio = 0.0434
samples = 900 0: 7 1: 2 2: 121 3: 471 4: 1 5: 194 6: 26 7: 32 8: 3 9: 43 class = 3
samples = 318 0: 65 2: 13 3: 22 4: 40 5: 22 6: 73 7: 3 8: 25 9: 55 class = 6
samples = 334 0: 44 2: 34 3: 16 4: 3 5: 17 6: 56 7: 1 8: 150 9: 13 class = 8
samples = 450 0: 112 2: 38 3: 32 4: 4 5: 29 6: 95 8: 39 9: 101 class = 0
samples = 681 0: 95 2: 67 3: 35 4: 2 5: 19 6: 100 7: 1 8: 331 9: 31 class = 8
small
medium
large
samples = 181 0: 23 2: 23 3: 45 5: 28 6: 21 7: 2 8: 22 9: 17 class = 3
slant
samples = 480 0: 37 2: 27 3: 124 5: 38 6: 7 8: 231 9: 16 class = 8
width::large <= 1.0 gini = 0.83 entropy = 2.8145 samples = 691 class = 2 info gain = 0.0653 gain ratio = 0.0653
samples = 203 0: 52 2: 19 3: 51 5: 26 6: 20 7: 2 8: 15 9: 18 class = 0
samples = 289 0: 14 1: 2 2: 31 3: 52 4: 5 5: 34 6: 25 7: 22 8: 3 9: 101 class = 9
samples = 209 0: 5 2: 26 3: 58 4: 2 5: 39 6: 21 7: 19 8: 3 9: 36 class = 3
small
medium
large
samples = 202 0: 6 2: 17 3: 62 5: 31 6: 18 8: 57 9: 11 class = 3
samples = 223 0: 34 2: 21 3: 55 5: 28 6: 27 8: 48 9: 10 class = 3
samples = 164 0: 23 2: 18 3: 38 5: 10 6: 16 8: 48 9: 11 class = 8
samples = 343 0: 34 2: 64 3: 26 4: 88 5: 46 6: 34 7: 24 8: 7 9: 20 class = 4
samples = 348 0: 51 2: 107 3: 7 4: 78 5: 14 6: 30 7: 50 8: 1 9: 10 class = 2
4
length
medium
large
height
small
medium
large
height
small
medium
large
length::medium <= 0.8 gini = 0.79 entropy = 2.6243 samples = 1829 class = 4 info gain = 0.0560 gain ratio = 0.0585
samples = 615 0: 28 1: 4 2: 145 3: 57 4: 75 5: 53 6: 45 7: 139 8: 7 9: 62 class = 2
samples = 241 0: 2 1: 1 2: 69 3: 49 5: 42 6: 6 7: 63 8: 1 9: 8 class = 2
slant
slant
slant
small
medium
large
small
medium
large
small
medium
large
samples = 1138 0: 37 1: 2 2: 241 3: 44 4: 478 5: 130 6: 28 7: 131 8: 6 9: 41 class = 4
length::large <= 0.9 gini = 0.85 entropy = 2.9297 samples = 848 class = 6 info gain = 0.1636 gain ratio = 0.1929
samples = 798 0: 286 2: 147 3: 30 4: 89 5: 61 6: 92 7: 7 8: 57 9: 29 class = 0
samples = 1141 0: 501 1: 1 2: 237 3: 64 4: 99 5: 64 6: 38 7: 7 8: 106 9: 24 class = 0
samples = 255 0: 48 2: 17 3: 23 4: 4 5: 23 6: 69 7: 2 8: 26 9: 43 class = 6
samples = 591 0: 232 2: 90 3: 46 4: 12 5: 47 6: 69 7: 3 8: 50 9: 42 class = 0
samples = 485 0: 236 2: 100 3: 38 4: 6 5: 10 6: 23 7: 3 8: 39 9: 30 class = 0
samples = 555 0: 76 2: 64 3: 177 5: 117 6: 46 8: 51 9: 24 class = 3
width::large <= 1.0 gini = 0.80 entropy = 2.5233 samples = 620 class = 0 info gain = 0.0614 gain ratio = 0.0623
samples = 296 0: 79 2: 77 3: 68 5: 29 6: 2 7: 3 8: 30 9: 8 class = 0
samples = 348 0: 51 2: 107 3: 7 4: 78 5: 14 6: 30 7: 50 8: 1 9: 10 class = 2
samples = 233 0: 11 2: 32 3: 14 4: 61 5: 8 6: 53 7: 15 8: 4 9: 35 class = 4
samples = 615 0: 137 2: 98 3: 56 4: 10 5: 48 6: 161 7: 7 8: 51 9: 47 class = 6
samples = 266 0: 44 2: 39 3: 86 4: 2 5: 39 6: 14 7: 1 8: 32 9: 9 class = 3
samples = 354 0: 110 2: 85 3: 63 5: 62 6: 7 7: 1 8: 23 9: 3 class = 0

Figure 12: **PART 2**: MCBM-Seq - Full Merged Tree on Morpho-MNIST with $msl = 150$, split in four images. The dark grey nodes show the decision rules of the global tree, and do not introduce leakage. The white grey nodes show the leaky splits of the corresponding sub-tree, if found. Indicators in the top of the pictures indicate where the following or previous picture starts. Indicator 1 shows the root of the tree.

A.9.1 DECISION PATHS

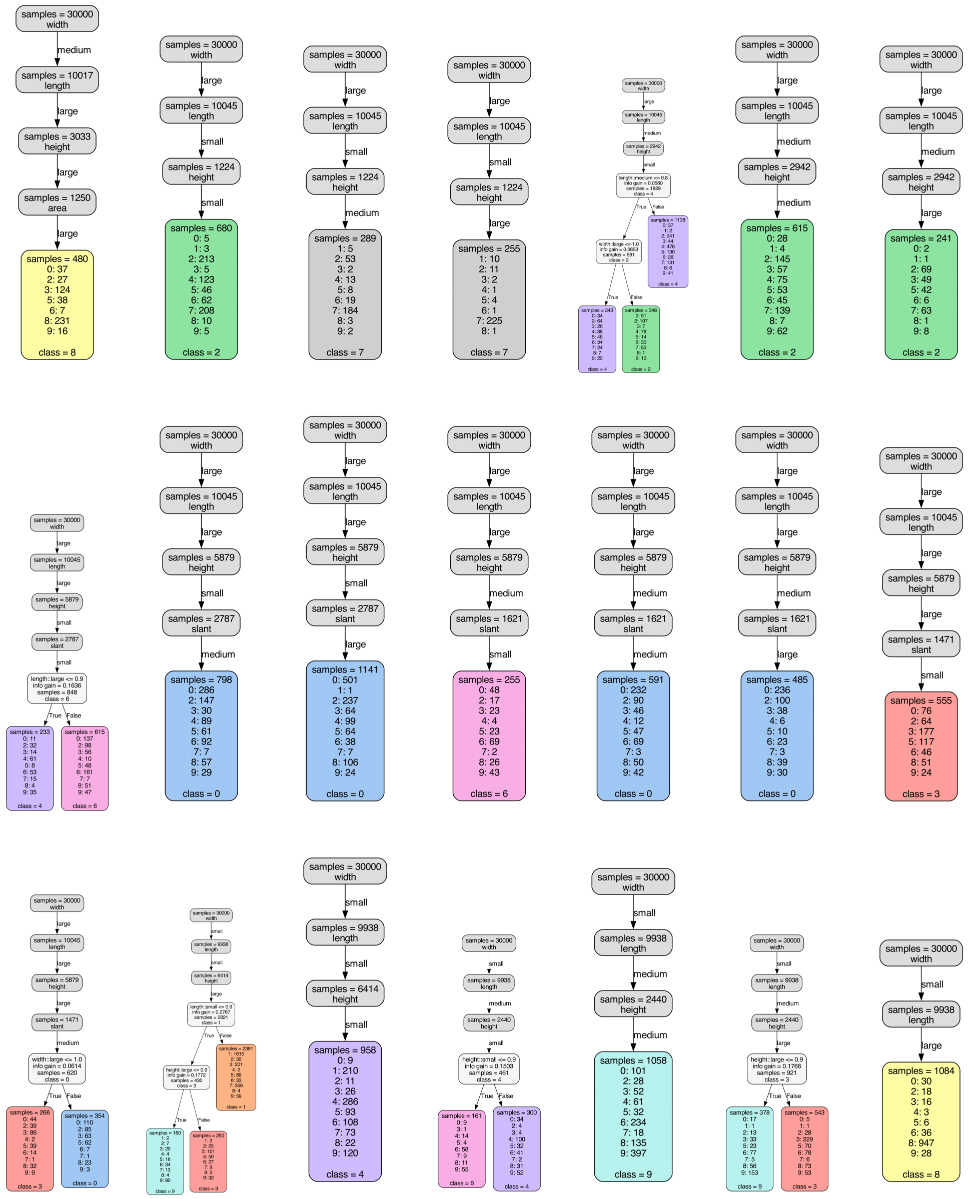


Figure 13: **PART 1**: MCBM-Seq - Full Merged Tree on Morpho-MNIST with $msl = 150$: All decision paths of the merged tree. If a sub-tree is found for a decision path, the complete sub-tree is shown.

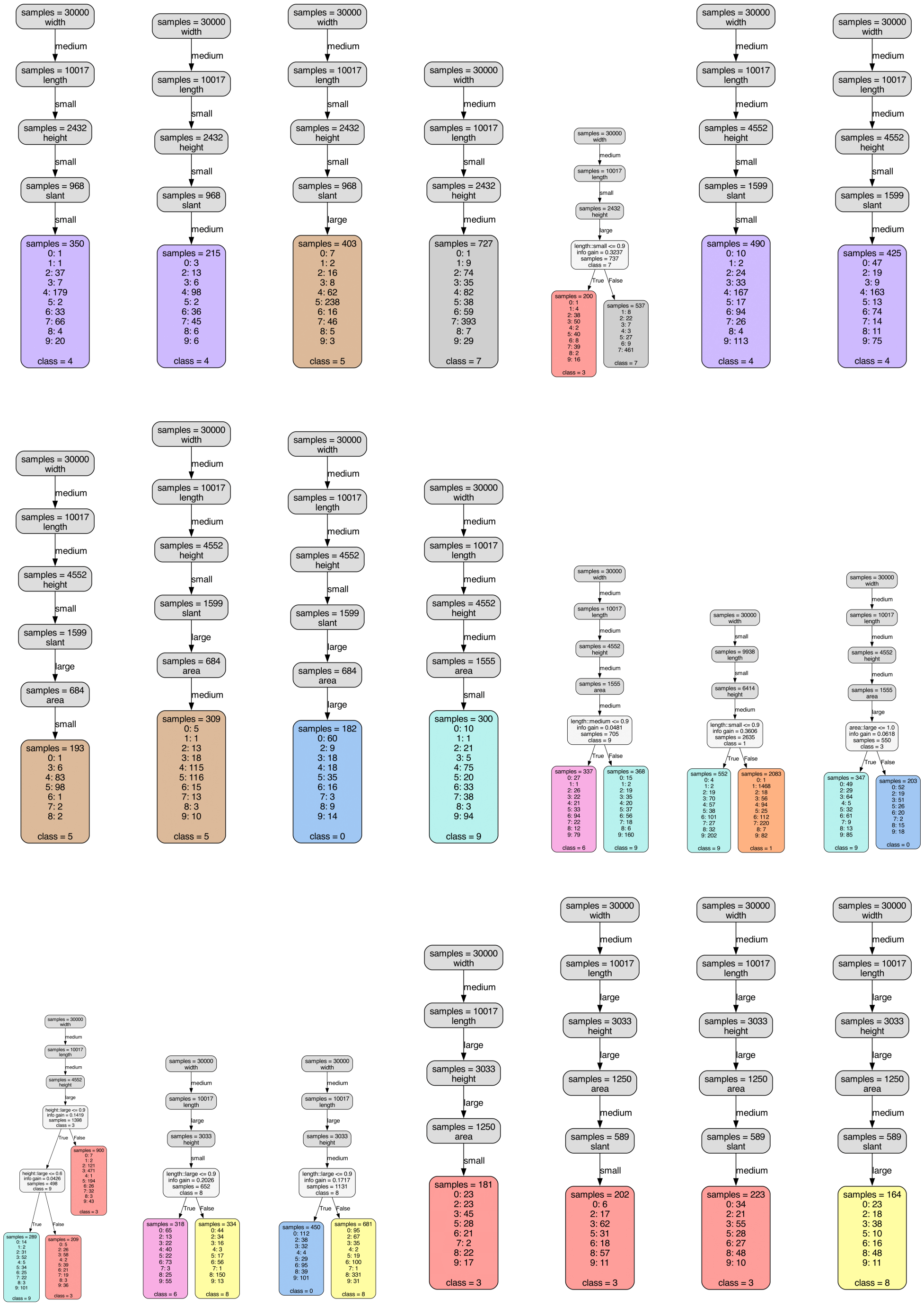


Figure 14: **PART 2**: MCBM-Seq - Full Merged Tree on Morpho-MNIST with $msl = 150$: All decision paths of the merged tree. If a sub-tree is found for a decision path, the complete sub-tree is shown.

A.10 CUB

We give the full decision path of the case study described in section 5.3 in Fig. 15 below. As we described, the label predictor observes that the concept predictor is **less than** 70% **confident** that the Red Bellied Woodpeckers have a solid breast pattern, while it is more confident for the vast majority of Red Headed Woodpeckers that they possess this attribute. While it is not the main focus of this work, we provide a visually plausible intuition for this result in Fig. 16, by examining many birds of the two classes. Our intuition is that the concept predictor generally assigns higher probabilities to Red Headed Woodpeckers because their breast colour is completely white and thus their pattern is solid, while that of Red Bellied Woodpeckers often shows orange dots. This is an example which shows that calibrated input probabilities can often be human-intuitive.

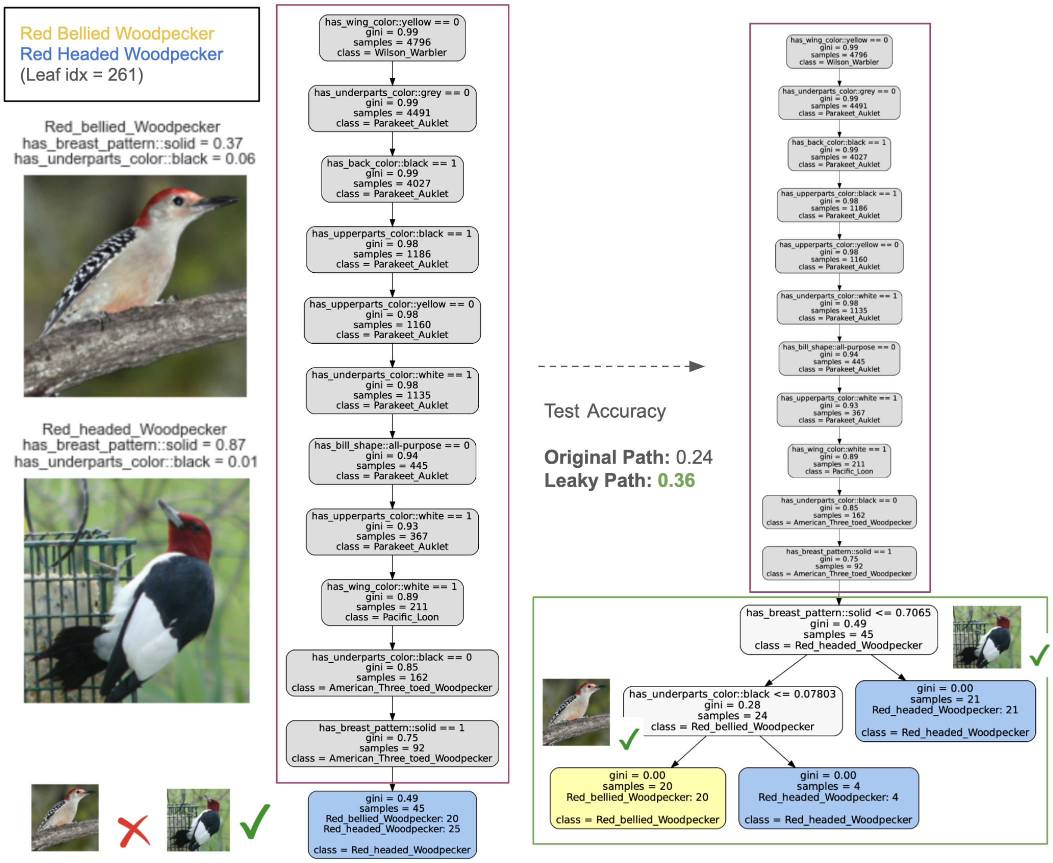

Red Bellied Woodpecker
Red Headed Woodpecker
(Leaf idx = 261)
Red_bellied_Woodpecker
has_breast_pattern::solid = 0.37
has_underparts_color::black = 0.06
Red_headed_Woodpecker
has_breast_pattern::solid = 0.87
has_underparts_color::black = 0.01
has_wing_color::yellow == 0 gini = 0.99 samples = 4796 class = Wilson_Warbler
has_underparts_color::grey == 0 gini = 0.99 samples = 4491 class = Parakeet_Auklet
has_back_color::black == 1 gini = 0.99 samples = 4027 class = Parakeet_Auklet
has_upperparts_color::black == 1 gini = 0.98 samples = 1186 class = Parakeet_Auklet
has_upperparts_color::yellow == 0 gini = 0.98 samples = 1160 class = Parakeet_Auklet
has_underparts_color::white == 1 gini = 0.98 samples = 1135 class = Parakeet_Auklet
has_bill_shape::all-purpose == 0 gini = 0.94 samples = 445 class = Parakeet_Auklet
has_upperparts_color::white == 1 gini = 0.93 samples = 367 class = Parakeet_Auklet
has_wing_color::white == 1 gini = 0.89 samples = 211 class = Pacific_Loon
has_underparts_color::black == 0 gini = 0.85 samples = 162 class = American_Three_toed_Woodpecker
has_breast_pattern::solid == 1 gini = 0.75 samples = 92 class = American_Three_toed_Woodpecker
gini = 0.49 samples = 45 Red_bellied_Woodpecker: 20 Red_headed_Woodpecker: 25 class = Red_headed_Woodpecker
✗
✓
Test Accuracy
Original Path: 0.24
Leaky Path: 0.36
has_wing_color::yellow == 0 gini = 0.99 samples = 4796 class = Wilson_Warbler
has_underparts_color::grey == 0 gini = 0.99 samples = 4491 class = Parakeet_Auklet
has_back_color::black == 1 gini = 0.99 samples = 4027 class = Parakeet_Auklet
has_upperparts_color::black == 1 gini = 0.98 samples = 1186 class = Parakeet_Auklet
has_upperparts_color::yellow == 0 gini = 0.98 samples = 1160 class = Parakeet_Auklet
has_underparts_color::white == 1 gini = 0.98 samples = 1135 class = Parakeet_Auklet
has_bill_shape::all-purpose == 0 gini = 0.94 samples = 445 class = Parakeet_Auklet
has_upperparts_color::white == 1 gini = 0.93 samples = 367 class = Parakeet_Auklet
has_wing_color::white == 1 gini = 0.89 samples = 211 class = Pacific_Loon
has_underparts_color::black == 0 gini = 0.85 samples = 162 class = American_Three_toed_Woodpecker
has_breast_pattern::solid == 1 gini = 0.75 samples = 92 class = American_Three_toed_Woodpecker
has_breast_pattern::solid <= 0.7065 gini = 0.49 samples = 45 class = Red_headed_Woodpecker
✓
has_underparts_color::black <= 0.07803 gini = 0.28 samples = 24 class = Red_bellied_Woodpecker
✓
gini = 0.00 samples = 21 Red_headed_Woodpecker: 21 class = Red_headed_Woodpecker
gini = 0.00 samples = 20 Red_bellied_Woodpecker: 20 class = Red_bellied_Woodpecker
gini = 0.00 samples = 4 Red_headed_Woodpecker: 4 class = Red_headed_Woodpecker

Figure 15: A case study in the CUB Wah et al. (2011) dataset. The Figure shows the corresponding decision path on the global tree (left) and how this is extended using Information Leakage (right) by the MCBM-Seq method to improve performance.

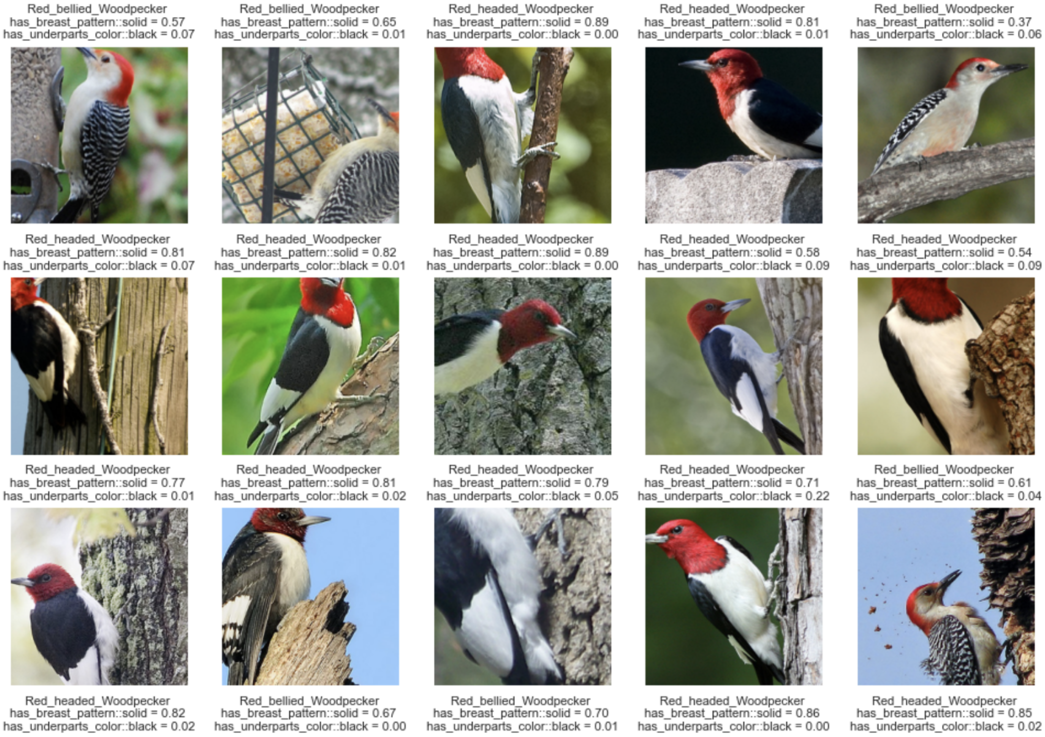

Red_bellied_Woodpecker
has_breast_pattern::solid = 0.57
has_underparts_color::black = 0.07
Red_bellied_Woodpecker
has_breast_pattern::solid = 0.65
has_underparts_color::black = 0.01
Red_headed_Woodpecker
has_breast_pattern::solid = 0.89
has_underparts_color::black = 0.00
Red_headed_Woodpecker
has_breast_pattern::solid = 0.81
has_underparts_color::black = 0.01
Red_bellied_Woodpecker
has_breast_pattern::solid = 0.37
has_underparts_color::black = 0.06
Red_headed_Woodpecker
has_breast_pattern::solid = 0.81
has_underparts_color::black = 0.07
Red_headed_Woodpecker
has_breast_pattern::solid = 0.82
has_underparts_color::black = 0.01
Red_headed_Woodpecker
has_breast_pattern::solid = 0.89
has_underparts_color::black = 0.00
Red_headed_Woodpecker
has_breast_pattern::solid = 0.58
has_underparts_color::black = 0.09
Red_headed_Woodpecker
has_breast_pattern::solid = 0.54
has_underparts_color::black = 0.09
Red_headed_Woodpecker
has_breast_pattern::solid = 0.77
has_underparts_color::black = 0.01
Red_headed_Woodpecker
has_breast_pattern::solid = 0.81
has_underparts_color::black = 0.02
Red_headed_Woodpecker
has_breast_pattern::solid = 0.79
has_underparts_color::black = 0.05
Red_headed_Woodpecker
has_breast_pattern::solid = 0.71
has_underparts_color::black = 0.22
Red_bellied_Woodpecker
has_breast_pattern::solid = 0.61
has_underparts_color::black = 0.04
Red_headed_Woodpecker
has_breast_pattern::solid = 0.82
has_underparts_color::black = 0.02
Red_bellied_Woodpecker
has_breast_pattern::solid = 0.67
has_underparts_color::black = 0.00
Red_bellied_Woodpecker
has_breast_pattern::solid = 0.70
has_underparts_color::black = 0.01
Red_headed_Woodpecker
has_breast_pattern::solid = 0.86
has_underparts_color::black = 0.00
Red_headed_Woodpecker
has_breast_pattern::solid = 0.85
has_underparts_color::black = 0.02

Figure 16: CUB: More birds of the classes ”Red bellied Woodpecker” and ”Red headed Woodpecker”. The confidence of the concept predictor is shown per bird for the concepts: ”has-breast-pattern-solid”, and ”has-underparts-colour-black”.

A.10.1 CUB:More Decision Paths

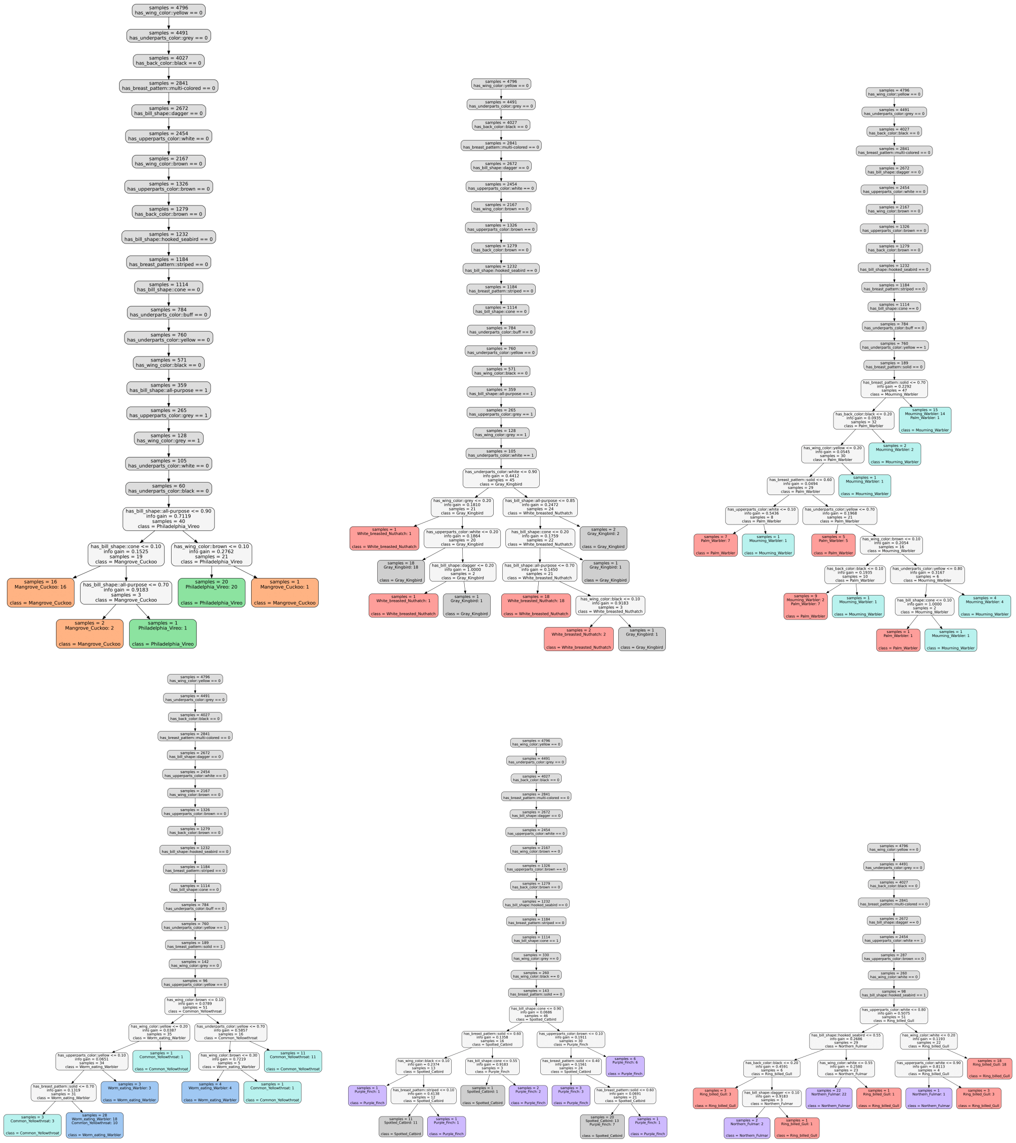

samples = 4796 has_wing_color::yellow == 0
samples = 4491 has_underparts_color::grey == 0
samples = 4027 has_back_color::black == 0
samples = 2841 has_breast_pattern::multi-colored == 0
samples = 2672 has_bill_shape::dagger == 0
samples = 2454 has_upperparts_color::white == 0
samples = 2167 has_wing_color::brown == 0
samples = 1326 has_upperparts_color::brown == 0
samples = 1279 has_back_color::brown == 0
samples = 1232 has_bill_shape::hooked_seabird == 0
samples = 1184 has_breast_pattern::striped == 0
samples = 1114 has_bill_shape::cone == 0
samples = 784 has_underparts_color::buff == 0
samples = 760 has_underparts_color::yellow == 0
samples = 571 has_wing_color::black == 0
samples = 359 has_bill_shape::all-purpose == 1
samples = 265 has_upperparts_color::grey == 1
samples = 128 has_wing_color::grey == 1
samples = 105 has_underparts_color::white == 0
samples = 60 has_underparts_color::black == 0
has_bill_shape::all-purpose <= 0.90 info gain = 0.7119 samples = 40 class = Philadelphia_Vireo
has_bill_shape::cone <= 0.10 info gain = 0.1525 samples = 19 class = Mangrove_Cuckoo
has_wing_color::brown <= 0.10 info gain = 0.2762 samples = 21 class = Philadelphia_Vireo
samples = 16 Mangrove_Cuckoo: 16 class = Mangrove_Cuckoo
has_bill_shape::all-purpose <= 0.70 info gain = 0.9183 samples = 3 class = Mangrove_Cuckoo
samples = 20 Philadelphia_Vireo: 20 class = Philadelphia_Vireo
samples = 1 Mangrove_Cuckoo: 1 class = Mangrove_Cuckoo
samples = 2 Mangrove_Cuckoo: 2 class = Mangrove_Cuckoo
samples = 1 Philadelphia_Vireo: 1 class = Philadelphia_Vireo
samples = 4796 has_wing_color::yellow == 0
samples = 4491 has_underparts_color::grey == 0
samples = 4027 has_back_color::black == 0
samples = 2841 has_breast_pattern::multi-colored == 0
samples = 2672 has_bill_shape::dagger == 0
samples = 2454 has_upperparts_color::white == 0
samples = 2167 has_wing_color::brown == 0
samples = 1326 has_upperparts_color::brown == 0
samples = 1279 has_back_color::brown == 0
samples = 1232 has_bill_shape::hooked_seabird == 0
samples = 1184 has_breast_pattern::striped == 0
samples = 1114 has_bill_shape::cone == 0
samples = 784 has_underparts_color::buff == 0
samples = 760 has_underparts_color::yellow == 0
samples = 571 has_wing_color::black == 0
samples = 359 has_bill_shape::all-purpose == 1
samples = 265 has_upperparts_color::grey == 1
samples = 128 has_wing_color::grey == 1
samples = 105 has_underparts_color::white == 1
has_underparts_color::white <= 0.90 info gain = 0.4412 samples = 45 class = Gray_Kingbird
has_wing_color::grey <= 0.20 info gain = 0.1810 samples = 21 class = Gray_Kingbird
has_bill_shape::all-purpose <= 0.85 info gain = 0.2472 samples = 24 class = White_breasted_Nuthatch
samples = 1 White_breasted_Nuthatch: 1 class = White_breasted_Nuthatch
has_upperparts_color::white <= 0.20 info gain = 0.1864 samples = 20 class = Gray_Kingbird
has_bill_shape::cone <= 0.20 info gain = 0.1759 samples = 22 class = White_breasted_Nuthatch
samples = 2 Gray_Kingbird: 2 class = Gray_Kingbird
samples = 18 Gray_Kingbird: 18 class = Gray_Kingbird
has_bill_shape::dagger <= 0.20 info gain = 1.0000 samples = 2 class = Gray_Kingbird
has_bill_shape::all-purpose <= 0.70 info gain = 0.1450 samples = 21 class = White_breasted_Nuthatch
samples = 1 Gray_Kingbird: 1 class = Gray_Kingbird
samples = 1 White_breasted_Nuthatch: 1 class = White_breasted_Nuthatch
samples = 1 Gray_Kingbird: 1 class = Gray_Kingbird
samples = 18 White_breasted_Nuthatch: 18 class = White_breasted_Nuthatch
has_wing_color::black <= 0.10 info gain = 0.9183 samples = 3 class = White_breasted_Nuthatch
samples = 2 White_breasted_Nuthatch: 2 class = White_breasted_Nuthatch
samples = 1 Gray_Kingbird: 1 class = Gray_Kingbird
samples = 4796 has_wing_color::yellow == 0
samples = 4491 has_underparts_color::grey == 0
samples = 4027 has_back_color::black == 0
samples = 2841 has_breast_pattern::multi-colored == 0
samples = 2672 has_bill_shape::dagger == 0
samples = 2454 has_upperparts_color::white == 0
samples = 2167 has_wing_color::brown == 0
samples = 1326 has_upperparts_color::brown == 0
samples = 1279 has_back_color::brown == 0
samples = 1232 has_bill_shape::hooked_seabird == 0
samples = 1184 has_breast_pattern::striped == 0
samples = 1114 has_bill_shape::cone == 0
samples = 784 has_underparts_color::buff == 0
samples = 760 has_underparts_color::yellow == 1
samples = 189 has_breast_pattern::solid == 0
has_breast_pattern::solid <= 0.70 info gain = 0.2292 samples = 47 class = Mourning_Warbler
has_back_color::black <= 0.20 info gain = 0.0935 samples = 32 class = Palm_Warbler
samples = 15 Mourning_Warbler: 14 Palm_Warbler: 1 class = Mourning_Warbler
has_wing_color::yellow <= 0.20 info gain = 0.0545 samples = 30 class = Palm_Warbler
samples = 2 Mourning_Warbler: 2 class = Mourning_Warbler
has_breast_pattern::solid <= 0.60 info gain = 0.0494 samples = 29 class = Palm_Warbler
samples = 1 Mourning_Warbler: 1 class = Mourning_Warbler
has_upperparts_color::white <= 0.10 info gain = 0.5436 samples = 8 class = Palm_Warbler
has_underparts_color::yellow <= 0.70 info gain = 0.1968 samples = 21 class = Palm_Warbler
samples = 7 Palm_Warbler: 7 class = Palm_Warbler
samples = 1 Mourning_Warbler: 1 class = Mourning_Warbler
samples = 5 Palm_Warbler: 5 class = Palm_Warbler
has_wing_color::brown <= 0.10 info gain = 0.2054 samples = 16 class = Mourning_Warbler
has_back_color::black <= 0.10 info gain = 0.1935 samples = 10 class = Palm_Warbler
has_underparts_color::yellow <= 0.80 info gain = 0.3167 samples = 6 class = Mourning_Warbler
samples = 9 Mourning_Warbler: 2 Palm_Warbler: 7 class = Palm_Warbler
samples = 1 Mourning_Warbler: 1 class = Mourning_Warbler
has_bill_shape::cone <= 0.10 info gain = 1.0000 samples = 2 class = Mourning_Warbler
samples = 4 Mourning_Warbler: 4 class = Mourning_Warbler
samples = 1 Palm_Warbler: 1 class = Palm_Warbler
samples = 1 Mourning_Warbler: 1 class = Mourning_Warbler
samples = 4796 has_wing_color::yellow == 0
samples = 4491 has_underparts_color::grey == 0
samples = 4027 has_back_color::black == 0
samples = 2841 has_breast_pattern::multi-colored == 0
samples = 2672 has_bill_shape::dagger == 0
samples = 2454 has_upperparts_color::white == 0
samples = 2167 has_wing_color::brown == 0
samples = 1326 has_upperparts_color::brown == 0
samples = 1279 has_back_color::brown == 0
samples = 1232 has_bill_shape::hooked_seabird == 0
samples = 1184 has_breast_pattern::striped == 0
samples = 1114 has_bill_shape::cone == 0
samples = 784 has_underparts_color::buff == 0
samples = 760 has_underparts_color::yellow == 1
samples = 189 has_breast_pattern::solid == 1
samples = 142 has_wing_color::grey == 0
samples = 96 has_upperparts_color::yellow == 0
has_wing_color::brown <= 0.10 info gain = 0.0789 samples = 51 class = Common_Yellowthroat
has_wing_color::yellow <= 0.20 info gain = 0.0387 samples = 35 class = Worm_eating_Warbler
has_underparts_color::yellow <= 0.70 info gain = 0.5857 samples = 16 class = Common_Yellowthroat
has_upperparts_color::yellow <= 0.10 info gain = 0.0651 samples = 34 class = Worm_eating_Warbler
samples = 1 Common_Yellowthroat: 1 class = Common_Yellowthroat
has_wing_color::brown <= 0.30 info gain = 0.7219 samples = 5 class = Worm_eating_Warbler
samples = 11 Common_Yellowthroat: 11 class = Common_Yellowthroat
has_breast_pattern::solid <= 0.70 info gain = 0.1319 samples = 31 class = Worm_eating_Warbler
samples = 3 Worm_eating_Warbler: 3 class = Worm_eating_Warbler
samples = 4 Worm_eating_Warbler: 4 class = Worm_eating_Warbler
samples = 1 Common_Yellowthroat: 1 class = Common_Yellowthroat
samples = 3 Common_Yellowthroat: 3 class = Common_Yellowthroat
samples = 28 Worm_eating_Warbler: 18 Common_Yellowthroat: 10 class = Worm_eating_Warbler
samples = 4796 has_wing_color::yellow == 0
samples = 4491 has_underparts_color::grey == 0
samples = 4027 has_back_color::black == 0
samples = 2841 has_breast_pattern::multi-colored == 0
samples = 2672 has_bill_shape::dagger == 0
samples = 2454 has_upperparts_color::white == 0
samples = 2167 has_wing_color::brown == 0
samples = 1326 has_upperparts_color::brown == 0
samples = 1279 has_back_color::brown == 0
samples = 1232 has_bill_shape::hooked_seabird == 0
samples = 1184 has_breast_pattern::striped == 0
samples = 1114 has_bill_shape::cone == 1
samples = 330 has_wing_color::grey == 0
samples = 260 has_wing_color::black == 0
samples = 143 has_breast_pattern::solid == 0
has_bill_shape::cone <= 0.90 info gain = 0.0686 samples = 46 class = Spotted_Catbird
has_breast_pattern::solid <= 0.60 info gain = 0.1358 samples = 16 class = Spotted_Catbird
has_upperparts_color::brown <= 0.10 info gain = 0.1911 samples = 30 class = Purple_Finch
has_wing_color::black <= 0.10 info gain = 0.2374 samples = 13 class = Spotted_Catbird
has_bill_shape::cone <= 0.55 info gain = 0.9183 samples = 3 class = Purple_Finch
has_breast_pattern::solid <= 0.40 info gain = 0.1561 samples = 24 class = Spotted_Catbird
samples = 6 Purple_Finch: 6 class = Purple_Finch
samples = 1 Purple_Finch: 1 class = Purple_Finch
has_breast_pattern::striped <= 0.10 info gain = 0.4138 samples = 12 class = Spotted_Catbird
samples = 1 Spotted_Catbird: 1 class = Spotted_Catbird
samples = 2 Purple_Finch: 2 class = Purple_Finch
samples = 3 Purple_Finch: 3 class = Purple_Finch
has_breast_pattern::solid <= 0.60 info gain = 0.0691 samples = 21 class = Spotted_Catbird
samples = 11 Spotted_Catbird: 11 class = Spotted_Catbird
samples = 1 Purple_Finch: 1 class = Purple_Finch
samples = 20 Spotted_Catbird: 13 Purple_Finch: 7 class = Spotted_Catbird
samples = 1 Purple_Finch: 1 class = Purple_Finch
samples = 4796 has_wing_color::yellow == 0
samples = 4491 has_underparts_color::grey == 0
samples = 4027 has_back_color::black == 0
samples = 2841 has_breast_pattern::multi-colored == 0
samples = 2672 has_bill_shape::dagger == 0
samples = 2454 has_upperparts_color::white == 1
samples = 287 has_upperparts_color::brown == 0
samples = 260 has_wing_color::white == 0
samples = 98 has_bill_shape::hooked_seabird == 1
has_upperparts_color::white <= 0.80 info gain = 0.5075 samples = 51 class = Ring_billed_Gull
has_bill_shape::hooked_seabird <= 0.55 info gain = 0.2686 samples = 29 class = Northern_Fulmar
has_wing_color::white <= 0.20 info gain = 0.1193 samples = 22 class = Ring_billed_Gull
has_back_color::black <= 0.20 info gain = 0.4591 samples = 6 class = Ring_billed_Gull
has_wing_color::white <= 0.55 info gain = 0.2580 samples = 23 class = Northern_Fulmar
has_upperparts_color::white <= 0.90 info gain = 0.8113 samples = 4 class = Ring_billed_Gull
samples = 18 Ring_billed_Gull: 18 class = Ring_billed_Gull
samples = 3 Ring_billed_Gull: 3 class = Ring_billed_Gull
has_bill_shape::dagger <= 0.10 info gain = 0.9183 samples = 3 class = Northern_Fulmar
samples = 22 Northern_Fulmar: 22 class = Northern_Fulmar
samples = 1 Ring_billed_Gull: 1 class = Ring_billed_Gull
samples = 1 Northern_Fulmar: 1 class = Northern_Fulmar
samples = 3 Ring_billed_Gull: 3 class = Ring_billed_Gull
samples = 2 Northern_Fulmar: 2 class = Northern_Fulmar
samples = 1 Ring_billed_Gull: 1 class = Ring_billed_Gull

Figure 17: Indicative Decision Paths of the merged MCBM-Seq tree on CUB. For the selected paths, a leaky sub-tree is found. Each new decision path distinguishes a pair of bird classes that were previously indistinguihable.