Leveraging Neurosymbolic AI for Slice Discovery

Michele Collevati [a], Thomas Eiter [a] and Nelson Higuera [a]
[a] *Institute of Logic and Computation, Technische Universität Wien, Vienna, Austria*
E-mails: michele.collevati@tuwien.ac.at, thomas.eiter@tuwien.ac.at, nelson.ruiz@tuwien.ac.at

Abstract.
While remarkable recent developments in deep neural networks have significantly contributed to advancing the state-of-the-art in Computer Vision (CV), several studies have also shown their limitations and defects. In particular, CV models often make systematic errors on important subsets of data called *slices*, which are groups of data sharing a set of attributes. A *slice discovery method* (SDM) is meant to detect semantically meaningful slices on which the model performs poorly, called *rare slices*. We propose a modular neurosymbolic SDM whose distinctive advantage is the extraction via inductive logic programming of human-readable logical rules describing rare slices, and thus enhancing the explainability of CV models. To this end, a methodology for inducing the occurrence of rare slices in a model is presented. We validate the SDM approach on both the synthetic *Super-CLEVR* and real-world *ImageNet* datasets. Our experiments demonstrate the complete pipeline: first, we successfully induce targeted rare slices using our taxonomy-based heuristic; second, our neurosymbolic SDM correctly identifies these slices and produces precise, human-readable logical rules to describe them; and finally, these rules are used to guide a data augmentation process that successfully mends model behaviour and improves its predictive performance.[1]

Keywords: Neurosymbolic AI, Slice Discovery, Inductive Logic Programming

1. Introduction

Computer Vision (CV) [56] is a field of Artificial Intelligence (AI) that enables computer systems to obtain semantic information from digital images and videos. Following the remarkable recent developments of deep neural networks, significant achievements have been made in advancing state-of-the-art performance in various CV tasks [31], among which it is crucial to mention safety-critical applications, such as autonomous driving [63].

However, empirical studies, e.g. [47], show that CV models struggle to generalise to new data slightly different from those on which they were initially trained and tested. A related problem is the presence of important subsets of data, called *slices*, for which deep learning models often make systematic errors [17]. A *slice* is defined as a group of data sharing a set of attributes. For instance, one study found that some object recognition models systematically underperform in identifying common household items from non-Western countries and low-income communities [14]. This underperformance likely stems from variations in the objects themselves and the different contexts in which they appear.

Accurately detecting underperforming slices, called *rare slices*, allows one to carefully analyse such prediction errors and subsequently improve the model. Expectedly, identifying rare slices is a complex

[1] The code for reproducing the experiments is available as an online repository: https://gitlab.tuwien.ac.at/kbs/nesy-ai/ilp4sd.

task, especially for high-dimensional and unstructured data, e.g. images, where such slices often manifest as subtle, non-obvious patterns that are difficult to spot and extract. Furthermore, it is non-trivial to understand what makes slices rare. In view of this, the *slice discovery problem* [17] has been described as mining unstructured input data for semantically meaningful slices on which the model performs poorly.

In this work, we propose to tackle the slice discovery problem with a *neurosymbolic* AI approach [23], given the capabilities of Machine Learning (ML), and in particular Deep Learning (DL), for unstructured data classification and Knowledge Representation and Reasoning (KRR) methods for transparent logical inference and explainability. In particular, we provide a framework to experiment with different datasets the effectiveness of our inductive logic programming-based *slice discovery method* (SDM). This framework allows us to evaluate our SDM according to the semantic quality of the extracted rules in describing rare slices and the effect of such rules in reducing model misclassifications. To this end, we leverage *Super-CLEVR* [36], a well-known synthetic dataset with a data generator for images of vehicles organised in hierarchical classes, and *ImageNet* [31], a large-scale real-world image dataset organised according to the WordNet [45] hierarchy. The main contributions of our work are summarised as follows:

1. We present our modular neurosymbolic framework for slice discovery, which consists of a closed-loop that involves data generation (or subsampling), object detection (or image classification), scene graph generation describing the semantic contents of images, rule learning to detect rare slices, and neural network model mending.
   We then translate the images classified by *YOLOv5* into scene graphs in the language of Inductive Logic Programming (ILP) [9]. Depending on the ground truth, these scene graphs constitute the positive and negative ILP examples, i.e. those in which the neural network incorrectly resp. correctly classified the image. Subsequently, we use three different ILP systems, *Popper* [10], *FOLD-R++* [59], and *FastLAS* [34], to obtain succinct logical rules that reveal which images are hard for the model to classify. Finally, the neural network model is trained on its checkpoint with further data generated using these rules.
2. In order to test the proposed approach on various slice discovery settings, we focus on generating datasets with rare slices. Closest to our work, Eyuboglu et al. [17] considered the generation of rare slices in the context of the hierarchical class structure, but did not consider further class taxonomies besides the default one; this makes their method not really suitable for the scenarios we are considering. In contrast, a taxonomy-based approach is pursued in this work, and a methodology for building datasets with rare slices is presented. We provide image datasets with rare slices leveraging *Super-CLEVR* (via generation) and *ImageNet* (via controlled subsampling), and on which we train *YOLOv5* [48] models.
3. We provide an implementation along with experimental results for both datasets to test the effectiveness of rare slice generation, rule extraction on the classification results of the neural network model, and model mending. The results show that our approach could reliably generate rare slices and that rule learning delivered meaningful rules describing rare slices. Furthermore, feeding the neural network with additional training data generated according to such rules resulted in a significant performance improvement, as misclassifications decreased considerably.

With our framework, we can generate controlled rare slices in datasets to then test the model behaviour on them. Furthermore, it allows the automatic mining via ILP of human-readable logical rules that pinpoint the deficiencies of a classification model and benefit the user's intuition for model mending. The transparent nature of logical rules makes them highly interpretable and provides a basis for finding model explanations from possible background information.

This article extends our previous work [7] with (i) a more detailed and extended related work section, (ii) the use of further ILP systems, (iii) a more rigorous experimental evaluation using an additional real-world dataset, and (iv) an improved implementation of the proposed SDM framework.

The remainder of the article is organised as follows. In Section 2, we provide a review of related work on SDMs. Section 3 presents an introduction to the *Super-CLEVR* and *ImageNet* datasets and the ILP

systems. Section 4 describes the proposed neurosymbolic framework for slice discovery. Section 5 presents a taxonomy-based methodology for generating datasets containing rare slices. Section 6 describes the experimental setup and presents an overview of the obtained results. The experimental results are discussed in Section 7. Finally, conclusions and future work are provided in Section 8.

2. Related Work

Several studies [5, 14, 28, 42] have shown that neural network models often make systematic errors on data slices. The impact of such errors is especially pronounced for critical application areas, such as medical diagnostics [43] and fraud detection [27], where accurate identification of rare slices positively influences essential decision-making. Consequently, recent research has proposed automated SDMs aimed at identifying semantically meaningful slices in which the model exhibits prediction errors. An optimal SDM should automatically detect data slices containing *coherent* instances that closely correspond to a concept understandable by humans [26] and on which the model *underperforms*.

Previous research has addressed the slice discovery problem by focusing on datasets with metadata or structured (e.g. tabular) data. In [6] the *Slice Finder* system is proposed, which employs two different automated data slicing methods, viz. decision tree training and lattice searching. In [49], the authors present *SliceLine*, an exact yet fast and practical enumeration algorithm to find problematic data slices leveraging monotonicity properties and upper bounds for effective pruning. On the other hand, the *Premise* algorithm [22] heuristically discovers those feature-value combinations (i.e. patterns) that provide clear insight into the systematic errors of NLP classifiers.

Dealing with the slice discovery problem becomes particularly challenging for unstructured data, such as images and audio. Recent studies have proposed methods for identifying slices in this context. Several of them embed the data in a representation space and then use clustering or dimensionality reduction techniques. The *Domino* SDM [17] exploits cross-modal embeddings and an error-aware Gaussian mixture model to discover and describe coherent slices, while the *Spotlight* method [13] for finding systematic errors is based on the idea that similar inputs tend to have similar representations in the final hidden layer of a neural network. *Spotlight* exploits this similarity by focusing on such representation space, aiming to identify contiguous regions where the model underperforms. In [55], the authors describe a two-step method, called *George*, for identifying underperforming slices without requiring access to slice labels. In the first step, slice labels are estimated by training a model and splitting each class into estimated slices through unsupervised clustering in the model feature space. In the second step, these estimated slices are used to train a new model, optimizing for worst-case performance over all estimated slices via a robust optimisation technique [50].

The recent explosion of generative AI has seen various works considering the use of such models to address the slice discovery problem. The *PromptAttack* procedure [40] identifies systematic errors by exploiting a text-to-image model to synthesise images, conditioned on a prompt that encodes information about subgroups and classes. In [21], a human-in-the-loop tool is proposed, called *AdaVision*, which uses GPT-3 [4] to suggest coherent but potentially underperforming slices to explore, and CLIP [46] to retrieve relevant images to improve slice identification. In [1], the authors present the *SCROD* pipeline for slice discovery in object detectors applied to synthetic street scenes. Such a pipeline consists of several generative models to synthesise images with fine-grained control in a fully automated and scalable way. The interactive *VLSlice* system [54] is designed to test vision-and-language models by discovering their slices from unlabelled image datasets. In [38], the *SSD-LLM* framework is proposed for automatic subpopulation structure discovery using a Large Language Model (LLM) [4]. Such a framework is based on the idea of generating informative image captions via a multimodal LLM [60], and then analysing and summarising the subpopulation structure of datasets through an LLM. *SSD-LLM* can be combined with subsequent operations to tackle various tasks better, including slice discovery.

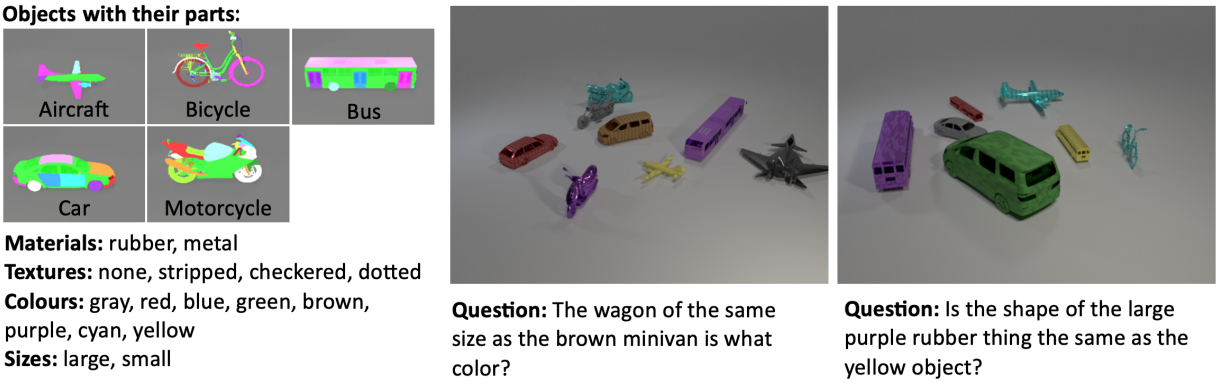

Fig. 1. The figure on the left shows images from *Super-CLEVR* [36] of vehicles made up of their parts characterised by four attributes, i.e. colour, size, material, and texture. The middle and right figures show examples of *Super-CLEVR* renderings with generated questions.

Distinguishing between positive and negative examples is central to our method. Prior work has leveraged Linear Temporal Logic over finite traces (LTLf) to separate temporal classes [19], and used Learning From Interpretation Transitions (LFIT) to explain black-box behaviour [57]. Other approaches generate interpretable Signal Temporal Logic (STL) formulas for time-series classification [61], integrate symbolic reasoning with neural models via abductive inference [12], or learn differentiable rule sets from continuous features [64]. Our method follows this line of research but focuses on ILP for discovering interpretable rules in the context of slice discovery for high-dimensional visual data.

While several prior works described above have tackled the problem of identifying data slices on which models underperform, they typically focus on black-box or subsymbolic techniques. A key challenge in this area is the lack of interpretability in the discovered slices. Our work directly addresses this gap by introducing a neurosymbolic approach that extracts human-readable logical rules to describe underperforming slices. Furthermore, we show that the rules can also be used effectively for mending the CV model, thus providing their direct practical application.

3. Preliminaries

In this section, we provide an overview of both the *Super-CLEVR* and *ImageNet* datasets and briefly overview ILP and the systems we use.

3.1. Super-CLEVR

Inspired by the seminal work on *CLEVR* [25], the *Super-CLEVR* dataset was designed to test the visual reasoning capabilities of AI systems. It comprises images featuring classes of vehicles, such as motorcycles, cars, and aeroplanes. The classes are further divided into vehicle subclasses, which make the dataset *hierarchical*, a crucial characteristic for inducing the occurrence of rare slices within our SDM framework. For example, the "motorcycle" class contains "chopper", "sportbike", "dirtbike", and "scooter" subclasses, also referred to as "shapes". The hierarchical structure of vehicle classes and their corresponding subclasses (shapes) defined in the original *Super-CLEVR* dataset are shown in Table 1. Vehicles have four attributes, i.e. colour, size, material, and texture. Each image is accompanied by a set of questions designed to test various aspects of visual reasoning, including types such as counting, existence, comparison, attribute identification, and spatial relationships as shown in Fig. 1. *Super-CLEVR* contains about $30k$ images and 10 question-answer pairs for each of them.

The *Super-CLEVR* dataset generator employs an algorithm that uses *Blender* [8] to create a diverse set of images and corresponding questions. Each image is generated by randomly placing vehicles in a

Table 1
Hierarchical structure of vehicle classes and their corresponding subclasses (shapes) defined in the original *Super-CLEVR* dataset.

Vehicle Class	**Vehicle Subclasses**
Aircraft	Private Jet, Fighter Jet, Biplane, Airliner
Bicycle	Road Bike, Mountain Bike, Utility Bike, Tandem Bike
Bus	Transit Bus, Double Bus, Articulated Bus, School Bus
Car	SUV, Pickup Truck, Station Wagon, Minivan, Sedan
Motorcycle	Dirtbike, Sportbike, Chopper, Scooter

three-dimensional scene. The attributes of these objects are also randomly assigned within predefined categories. Spatial relationships are managed to ensure objects do not overlap unrealistically. Once an image is composed, the generator creates questions based on different types of reasoning tasks. The questions are formulated by randomly selecting objects and their attributes in the image and constructing queries that require an understanding of the objects and their relations. This procedural generation ensures a wide variety of questions and scenes, overcoming possible human biases when creating datasets.

Example 1 (*running example*)**.** Throughout the paper, we use a running example to illustrate our methodology: a rare slice from the *Super-CLEVR* dataset involving the "utility bike" subclass. This slice is statistically rare, appearing with a very low occurrence frequency in the training data, and it is visually similar to the "mountain bike" subclass, making it a candidate for model misclassification and an ideal test case for the proposed SDM pipeline.

3.2. ImageNet

To validate our SDM framework on a real-world benchmark, we use the well-known *ImageNet* [31] dataset. Unlike the synthetic *Super-CLEVR*, *ImageNet* is a large-scale image dataset consisting of real-world images organised according to the WordNet [45] hierarchy. It contains over 14 million annotated images representing more than 20,000 categories, making it one of the most widely used benchmarks in CV. For our experiments, we did not use the entire, vast WordNet hierarchy. Instead, to create a realistic but controlled setting for evaluating the proposed SDM, we defined a custom taxonomy based on a curated subset of vehicles from WordNet. This allowed us to apply our taxonomy-based heuristic to induce the generation of specific rare slices in a focused and challenging experimental environment to test our methodology.

ImageNet is primarily designed for *image classification*, where each image is associated with a single class label corresponding to the main object in the scene. It does not provide the detailed, object-level bounding box annotations found in object detection datasets. Furthermore, as a dataset of real-world images, *ImageNet* lacks a procedural image generator. Therefore, creating rare slices or augmenting the data for model mending are achieved through controlled subsampling of the existing dataset or using external data augmentation techniques.

3.3. Inductive Logic Programming

Inductive Logic Programming [41] is a subfield at the intersection of ML and KRR that aims to find patterns in data by learning logical descriptions, utilising background knowledge (B) and sets of positive (E^+) and negative (E^-) ground examples. The learning process in ILP aims to find a hypothesis h from a hypothesis space H, such that $B \cup h \models E^+$, and $B \cup h \not\models E^-$, i.e. the background B plus the hypothesis h

entails each positive example while it does not entail any negative example. It typically involves, starting from the known facts and relations contained in B, generating hypotheses consistent with E^+, testing them against E^- to ensure that no negative example is entailed, and refining them until they entail all the positive and none of the negative examples. If no hypothesis satisfies this condition, the learning process ends with no solution. While classical ILP systems often assume a single, global background knowledge B shared across all examples, our framework operates on *context-dependent* examples. In this setting, each example is accompanied by its own scene-specific background knowledge derived from the corresponding image.

ILP has applications in various fields, among them robotics [62], bioinformatics, e.g. protein structure discovery [58], medicine, e.g. drug design [15, 18], and ECG waveform learning [29], to mention a few; see [3, 32] for more of them.

A number of ILP approaches and tools are available; for a comprehensive survey on ILP, we refer to [11]. This work tests the following three ILP systems as symbolic reasoning components within the proposed neurosymbolic architecture for slice discovery:

1. *Popper* [10] is a state-of-the-art first-order ILP system that implements the learning from failures approach by combining Answer Set Programming (ASP) [37] and Prolog [2]. It supports infinite problem domains, reasoning about lists and numbers, learning textually minimal programs, and learning recursive programs. Furthermore, *Popper* can learn minimal description length logic programs as hypotheses from noisy data.
2. *FOLD-R++* [59] is in terms of efficiency and scalability an improvement of the *FOLD-R* first-order inductive learning algorithm [52], which serves to learn answer set programs from mixed (numerical and categorical) data for classification tasks. The three main improvements of *FOLD-R++* are the following: (i) it uses the prefix sum algorithm to speed up computation; (ii) it allows negated literals in the default portion of the learnt rules; (iii) it introduces the hyper-parameter *exception ratio*, which is the threshold of the ratio of false-positive examples (i.e. exceptions) to true-positive examples that a rule can imply.
3. *FastLAS* [34] is a recent first-order ILP system designed to perform learning tasks in the context of ASP, based on the context-dependent learning-from-answer-sets framework used by the *ILASP* [33] system. *FastLAS* comes with several restrictions, i.e. it is not as general as *ILASP*, but it is significantly more scalable. Furthermore, *FastLAS* has the advantage of taking as input a customised scoring function for hypotheses that allows the user to express domain-specific optimisation criteria. Such a scoring function defines the cost of a rule. *FastLAS* then computes an optimal solution with respect to the given scoring function. Finally, a key feature of *FastLAS* is its capability to handle noisy data by introducing a penalty mechanism. This mechanism assigns a penalty to each example, which is a cost for not covering that example. The penalties are defined by a user-specified weight, denoted by the variable $\lambda > 0, \lambda \in \mathbb{N}$, which adjusts the importance of different examples. Higher penalties may discourage the system from including certain complex rules if simpler alternatives exist, thus balancing accuracy against simplicity in the learned model.

The ILP systems we use, with the exception of *FOLD-R++*, are guided by a *mode bias*, which is a set of declarations that constrains the structure of the hypotheses the system can learn. These declarations specify, for example, which predicates can appear in the head or body of a rule, their argument types, and whether negation is permitted. This syntactic bias is crucial for pruning the vast hypothesis space and focusing the search on meaningful rule templates.

The ability of these three ILP systems to learn from noisy data is fundamental to the functioning of our SDM. Indeed, a rare slice can be interpreted as a set of exceptions on which a classifier underperforms. In order to find the pattern that characterises such a set of exceptions, we use the same idea as in [52], which is to consider misclassifications as positive examples and correct classifications as negative examples to obtain rules describing rare slices.

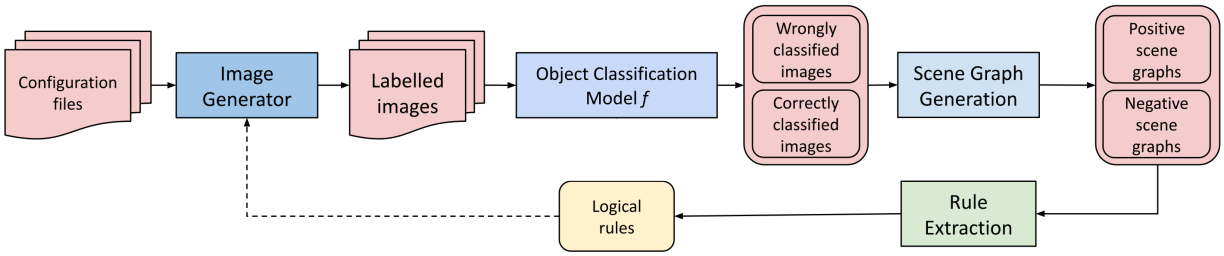

Fig. 2. Overview of the proposed neurosymbolic SDM architecture. The solid arrows show the data flow, while the dashed arrow represents the use of the extracted logical rules to generate further training data to improve classification performance.

Example 2 (*continued*). Continuing our example, suppose the trained model frequently confuses "utility bike" with "mountain bike" and misclassifies it as "sports bicycle" instead of "urban bicycle". After feeding these misclassifications (positive examples) and correct classifications (negative examples) into an ILP system, it might produce the following logical rule:

```
hard(V0) :- contains(V0, V1), shape(V1, utility_bike), color(V1, yellow),
            material(V1, rubber), direction(V1, south), sce_id(V0), obj_id(V1).
```

This rule provides a precise, human-readable diagnosis. It has learned that "a scene `V0` is hard for the model to classify if it contains an object `V1` whose shape is utility bike, colour is yellow, material is rubber, and direction is south". The `sce_id(V0)` and `obj_id(V1)` predicates simply bind the variables to the scene and object identifiers, respectively. This symbolic output is the key to understanding the rare slice and is used to guide the subsequent model mending process.

4. Neurosymbolic Framework for Slice Discovery

In order to construct a neurosymbolic SDM approach, we propose an architecture of a system as shown in Fig. 2. The system comprises several modules, shown as boxes, which process inputs in a pipeline. From configuration files or available data sources, datasets containing rare slices are constructed (either through generation or subsampling) on which a neural network model is trained and evaluated. Then, a semantic description of the images is produced, from which rules for detecting rare slices are extracted. Finally, the rules are used to generate further training data to mend the neural network model, thus closing the loop of model learning. In the following, we describe the tasks in the processing pipeline in more detail.

According to Kautz's taxonomy of neuro-symbolic systems [51], our SDM approach aligns with the [*Neuro→Symbolic*] paradigm, where the outputs of a neural system (here, a vision model) are post-processed by a symbolic module to derive interpretable logical rules.

4.1. Data Generation

The first step in the pipeline is concerned with data generation, i.e. producing datasets containing rare slices. We provide a methodology for them, which will be detailed in Section 5.

CV encompasses a wide range of tasks, among which image classification and object detection are two of the most prominent. Image classification assigns a single label to an entire image, identifying the most prominent object or scene. In contrast, object detection involves identifying and localising multiple objects within an image by predicting both their classes and bounding boxes. Our framework handles both tasks but requires different processing pipelines accordingly.

At an abstract level, the task consists of creating a labelled dataset D, where elements are labelled with their ground-truth annotations. In our general framework, D consists of pairs (I, L), where I is an image

and L is its label . For *Super-CLEVR*, L includes both the class and bounding box annotations, whereas for *ImageNet*, L contains only the class label.

For illustration, in the *Super-CLEVR* setting $L = \{(b_1, h_1), \ldots, (b_n, h_n)\}$, where each b_i is a bounding box of some object o_i in I, and h_i is in the underlying hierarchy $\mathcal{H}$ the root class of the subclass of o_i, $1 \leqslant i \leqslant n$. For example, the vehicle subclass (or "shape") "dirtbike" may have as its root class "land vehicle", "motorcycle", or another class depending on the hierarchy $\mathcal{H}$ under consideration. A bounding box b_i is a tuple (x^-, y^-, x^+, y^+), where (x^-, y^-) is the top-left corner point and (x^+, y^+) is the bottom-right corner point. Objects are identified by their bounding boxes, i.e. we can view b_i as an object ID. Following the *ImageNet* standard, the dataset provides a single class label L for each image. It does not include the bounding boxes or scene graphs, so additional annotations can be generated using external tools.

For datasets with a synthetic generator, such as *Super-CLEVR*, we directly modify the generator that renders images to control the distribution of objects. By adjusting the occurrence frequency of specific objects according to a given hierarchy $\mathcal{H}$, we can create a new dataset that contains controlled rare slices. The dataset is split into a training and a validation set, based on information provided in configuration files, e.g. whether or not each split contains rare slices. The generator produces a labelled dataset $D = \{(I_1, L_1), \ldots, (I_{N_s}, L_{N_s})\}$ where the number of images N_s per split s is approximately n_s/β, with n_s being the total number of objects per split and β the average number of objects per image. For real-world datasets where no generator is available, such as *ImageNet*, we simulate this process. We achieve the same outcome by performing a controlled subsampling of the original dataset to construct new training and validation splits with the desired distribution of rare slices.

The *Super-CLEVR* generator produces further data for the images, such as questions about them (which we disregard, as not needed) and scene descriptions consisting of object attributes (e.g. colour and size). This enriched description is the ground truth of the images, which can be used for synthetic scene graph generation and fine-grained classification. For *ImageNet*, which lacks such built-in annotations, we obtain comparable semantic descriptions through automated scene graph generation methods applied to the subsampled images, as later described in Section 6.3.1.

4.2. Object Detection and Image Classification

Once a dataset is prepared, we train and evaluate a neural network model to produce the classification results that will be analysed for rare slices discovery. This process involves a standard training and validation cycle, followed by a specific step to categorise the results for our SDM pipeline. First, a model (e.g. *YOLOv5*) is trained on the training split of the dataset D, which contains the induced rare slices. The trained model is then run on the validation split, where the object distribution is balanced to fairly evaluate the performance of the model. The model prediction format depends on the CV task associated with each dataset. For image classification (e.g., on *ImageNet*), the model returns a single class label $\hat{h}$ for a given image I. For object detection (e.g., on *Super-CLEVR*), it returns a set of pairs $(\hat{b}, \hat{h})$, where each pair consists of a predicted bounding box $\hat{b}$ and its corresponding class label $\hat{h}$ from the hierarchy $\mathcal{H}$ with its associated confidence score. Finally, to prepare the data for rule extraction, we analyse the model performance on the validation set and, for each class $h \in \mathcal{H}$, partition the validation images into two sets:

- E_h^+ *(positive examples)*: This set contains images where the model failed. For object detection, this means any image where at least one object of class h was misclassified. For image classification, it is any image of class h that received an incorrect label.
- E_h^- *(negative examples)*: This set contains images where the model succeeded. For object detection, this means all objects of class h in the image were correctly classified. For image classification, it is any image of class h that was correctly labelled.

These sets of positive (failure) and negative (success) examples constitute the input for the subsequent scene graph generation and rule extraction steps.

4.3. Scene Graph Generation

Scene Graph Generation (SGG) deals with producing a semantic graph representation from an input image. A scene graph is a (labelled) directed graph $G = (V, E)$ comprising object nodes, attribute nodes, and relation nodes. Each object is typically associated with a bounding box, a class label, and attributes such as colour or size. Relations capture connections between object pairs, e.g. spatial relations like *behind* or *next to*. Scene graphs offer a powerful semantic abstraction of visual data that can be leveraged to identify patterns linked to misclassifications and support interpretable rule extraction. Various SGG methods exist, most based on deep neural networks, but symbolic or hybrid approaches are also possible.

In synthetic datasets like *Super-CLEVR*, ground-truth scene graphs can be directly obtained from the dataset generator, as rich annotations are available for all objects, attributes, and relations. These ground-truth graphs provide a perfect basis for constructing logical examples for rule extraction. In contrast, for real-world datasets like *ImageNet*, which lack such ground-truth annotations, scene graphs must be generated through external tools (e.g., automated SGG methods or additional annotation pipelines [35]). These tools are able to derive the necessary object nodes, attribute nodes, and relation nodes, the latter also deducible from spatial configurations. In both settings, we convert validation examples in E_h^+ and E_h^- into scene graphs G_{E^+} and G_{E^-}, respectively: for *Super-CLEVR*, these are constructed using ground-truth annotations; for *ImageNet*, they are generated using additional tools as described in Section 6. Regardless of the source, each image in the validation set is converted into this structured graph format, ensuring a consistent semantic representation suitable for the subsequent ILP-based rule extraction step.

4.4. Rule Extraction via Inductive Logic Programming

We define an instance of the *rule extraction problem* in the ILP language to detect rare slices. To this end, we translate the scene graphs in $G_{E_h^+}$ resp. $G_{E_h^-}$ of class h into their logical representations which are suitable for assembling the sets $ILP_{E_h^+}$ and $ILP_{E_h^-}$ of positive and negative examples, respectively, and the background knowledge B describing the semantic information about objects in the images. This involves converting the objects and their attributes depicted in the scene graphs into logical facts.

Context-Dependent Background Knowledge. We clarify that, while classical ILP systems often assume a single, global background knowledge B shared across all examples, in our setting the examples are *context-dependent*: each example includes its own scene-specific information derived from the corresponding image. This is the case for *FastLAS*, where the background facts describing object attributes are included directly in each example (see Figure 3). For *Popper*, the background knowledge is provided separately from the examples, but it is still paired uniquely with each example via identifiers. In the case of *FOLD-R++*, the background knowledge is explicit in the tabular representation defined by CSV columns as features. Formally, for a set of examples E, for each $e_i \in E$ there exists a distinct background knowledge B_i specifically related to e_i. For ease of notation, in the rest of the paper, we will refer to this context-dependent background knowledge generally as B. Thus, while the notion of B is preserved, it is virtually instantiated per-example in different forms across the ILP systems we evaluate.

Then, $ILP_{E_h^+}$, $ILP_{E_h^-}$, and B are fed into a rule extraction system. Notably, the positive examples $ILP_{E_h^+}$ represent the input images for which the model made an incorrect classification, as we look for an explanation of why the model fails. The rule extraction system, for which we envisage using an ILP system, then outputs a set of rules as a hypothesis for rare slice detection.

As an example of ILP encoding, Fig. 3 shows excerpts of positive and negative examples and part of the mode bias used for *Super-CLEVR*. The representation is in the language of *FastLAS*, a state-of-the-art ILP system. Its expressive language allows the description of data and the specification of parameters and mode declarations to shape the search space. Specifically, *FastLAS* allows for a penalty to be set for each example by coding `sX@Y`, where X is a scene ID and Y is the cost for not covering that example. The positive example, denoted by `#pos`, is entailed by its background knowledge B, which is the third set `{contains(19, 0)...}` listed, combined with the hypothesis h if there is at least one answer set that

```
#pos(s19@4, {hard(19)}, {}, {
    contains(19, 0).
    color(0, yellow).
    direction(0, south).
    material(0, metal).
    shape(0, utility).
    size(0, large).

    contains(19, 1).
    ...
}).


#neg(s32@2, {hard(32)}, {}, {
    contains(32, 0).
    color(0, gray).
    direction(0, southwest).
    material(0, rubber).
    shape(0, tandem).
    size(0, small).

    contains(32, 1).
    ...
}).
```

```
% Configuration options
#maxv(2).

% Rule heads
#modeh(hard(var(sce_id))).
% Rule bodies
#modeb(contains(var(sce_id), var(obj_id))).
#modeb(not contains(var(sce_id), var(obj_id))).
#modeb(shape(var(obj_id), const(shape))).
#modeb(color(var(obj_id), const(color))).
#modeb(size(var(obj_id), const(size))).
#modeb(not size(var(obj_id), const(size))).
#modeb(direction(var(obj_id), const(direction))).
#modeb(not direction(var(obj_id), const(direction))).
#modeb(material(var(obj_id), const(material))).
#modeb(not material(var(obj_id), const(material))).

% Type domains
shape(utility).
direction(east).
...

% Scoring function
#bias("penalty(20, head(X)) :- in_head(X).").
```

Fig. 3. The left side of the figure shows excerpts of positive and negative examples with their context-dependent background knowledge, in the language of *FastLAS*. The right side of the figure shows an excerpt of the mode bias.

includes the ground atom `hard(19)`. The negative example, denoted by `#neg`, is not entailed if there is no answer set of its background knowledge B combined with h that includes the `hard(32)` atom. Background knowledge B for each example is derived from the scene graphs of the images. The `hard/1` predicate was explicitly introduced as a rule head to represent in which case a scene is difficult for the classifier, i.e. it contains rare slices, depending on its composition of objects and attributes. The mode bias specifies that the `hard(X)` predicate must only appear as a rule head, where X is a scene ID. Conversely, the `contains(X,Z)` predicate, where X is a scene ID and Z is an object ID, can only appear in the rule body. The same applies to the `shape(Z,sha)`, `color(Z,col)`, `size(Z,siz)`, `direction(Z,dir)`, and `material(Z,mat)` predicates, where Z is an object ID and *sha*, *col*, *siz*, *dir*, and *mat* are constants of the respective domains. Furthermore, four predicates, i.e. `contains`, `size`, `direction`, and `material`, can also appear as negative literals in the rule body. In the *FastLAS* mode bias, only a subset of predicates is specified as negative to tailor the search space. Finally, *FastLAS* allows the specification of the maximum number of variables per rule via the `#maxv` directive, and the scoring function via `#bias("penalty...")`.

Example 3 (*continued*)**.** Consider a scene where "a yellow, rubber utility bike facing south" is misclassified by the object detector. Its corresponding scene graph is translated into a positive example for the ILP system for rule extraction. In the *FastLAS* syntax, each `#pos` or `#neg` block defines one example. For instance, in the positive example `#pos(s19@4, {hard(19)}, {}, {contains(19, 0). ...}).` in Fig. 3, the first argument is the unique example identifier plus its penalty, the second is the set of atoms that the learned rules must prove (in this case, `hard(19)`), the third specifies atoms the rules must not prove (unused in our examples), and the fourth block contains the context-dependent background knowledge, a set of facts describing this specific scene. In this case, the fact `contains(19, 0)` links object 0 to scene 19, while the subsequent facts, such as `shape(0, utility)` and `color(0, yellow)`, define its attributes. In contrast, a scene containing a correctly classified vehicle, like a "tandem bike" in Fig. 3, would be added as a negative example `#neg(...).`.

The mode bias then defines the structure for the rules the ILP system can learn. For example, `#modeh(hard(var(sce_id))).` declares that the head of any learned rule must be of the form

`hard(SceneID)`, identifying difficult scenes for the model, while `#modeb` lists all admissible object predicates and their values for the rule body.

Given the complete input in Fig. 3, *FastLAS* produces the following hypothesis h:

```
hard(V0) :- contains(V0,V1), size(V1,large), material(V1,rubber), shape(V1,utility),
            direction(V1,south), sce_id(V0), obj_id(V1).
hard(V0) :- contains(V0,V1), shape(V1,utility), color(V1,yellow), direction(V1,south),
            sce_id(V0), obj_id(V1).
hard(V0) :- contains(V0,V1), shape(V1,utility), direction(V1,north), sce_id(V0), obj_id(V1).
hard(V0) :- not size(V1,large), not direction(V1,east), not direction(V1,southeast),
            contains(V0,V1), shape(V1,utility), color(V1,purple), sce_id(V0), obj_id(V1).
```

Informally, these rules express that a scene is considered difficult for the model to classify if it contains a "utility bike" with specific attributes. For example, the first rule says that whenever there is a scene with a "large rubber utility bike facing south", the neural network model will likely make a misclassification error on such an object.

The *Popper* encoding is very similar in structure to the *FastLAS* encoding since it consists of a file for specifying the examples, one for the background knowledge, and one for the mode bias. In contrast, the *FOLD-R++* encoding is more simplified as it only consists of a CSV file of tabular data, where the first row specifies the feature names for each column, and subsequent rows provide all the examples. The complete encodings for the *Popper*, *FOLD-R++*, and *FastLAS* systems used in the experiments are available in the online repository, with excerpts provided in Appendix B.

4.5. Model Mending

The final step in our SDM pipeline is model mending, where the extracted rules discovered in the previous stage are used to correct model deficiencies. This is achieved by augmenting the original training data with new images that specifically target the identified rare slices. For a synthetic dataset like *Super-CLEVR*, we use the data generator to procedurally create new images that precisely match the conditions specified by the logical rules. For a real-world dataset like *ImageNet*, while generative models offer one possible path for data augmentation, our approach instead relies on curated subsampling. We identify and select images from the complete *ImageNet* dataset that match the rule conditions to effectively augment the training data without generating synthetic images. In both cases, the model is then retrained on this enriched dataset to improve its robustness and performance on rare slices.

Example 4 (*continued*)**.** Fig. 5 shows the effectiveness of our model mending process: before the intervention, the model misclassifies the "utility bike" rare slice as "sports bicycle", whereas after retraining it with data guided by the extracted rules, it correctly classifies it as "urban bicycle".

5. Rare Slice Generation Methodology

Motivated by the limitations of existing rare slice generation methods in our context and the need for a reliable testbed for our SDM, we present a taxonomy-based methodology to induce the occurrence of controlled rare slices in a model.

In our framework, following the characterisation introduced by Domino [17], we define a rare slice as an object subclass that appears infrequently in the dataset and on which the model underperforms. Therefore, a rare slice has two key properties:

1. *Statistical Rarity*: The slice appears with a very low frequency in the dataset.
2. *Functional Rarity*: The model systematically underperforms on the slice.

While statistical rarity can be directly controlled (e.g., by setting a low occurrence probability via the *Super-CLEVR* generator, or via controlled subsampling of the *ImageNet* dataset), functional rarity is a model-dependent property. To reliably induce functional rarity, we propose a heuristic that bases its intuition on the fact that a CV model is more likely to fail when forced to distinguish between visually similar objects that belong to different target classes. Therefore, our heuristic is to intentionally design custom taxonomies that separate these similar object subclasses into distinct target classes (e.g., placing "dirtbike" in the "motorcycle" class and the visually similar "mountain bike" in the "bicycle" class). By then making one of these subclasses statistically rare, we induce the generation of a controlled rare slice in the classification model. This approach can be applied to any dataset, including *Super-CLEVR* (by defining custom class hierarchies for the generator) and *ImageNet* (by grouping and re-mapping classes from its native WordNet hierarchy).

To formally identify these underperforming slices, we use a main performance metric for each task as a proxy for difficulty. A target class is flagged as possibly containing a rare slice if its metric falls below a dataset-dependent target class threshold τ_c. Specifically, we use per-class *recall* for the *Super-CLEVR* object detection task and *Top-1 accuracy* for the *ImageNet* image classification task. This threshold allows us to systematically identify underperforming slices and study them in controlled settings.

Our methodology for generating rare slices begins with a given class hierarchy $\mathcal{H} = \{h_1 : \bar{s}_1, \ldots, h_m : \bar{s}_m\}$, where each root class h_i contains a set of subclasses $\bar{s}_i$. We then identify a set of pairs $P = \{(c_1, c_2), \ldots, (c_{n-1}, c_n)\}, n \geqslant 1$, of visually similar subclasses, each belonging to some $\bar{s}_i$, that a CV model is likely to confuse (e.g., "mountain bike" and "dirtbike"). Finally, we construct a dataset D such that, for a given class $h_i \in \mathcal{H}$, one of its subclasses c_i appearing in a pair of P occurs with a very low occurrence probability α in D. We implement this methodology through a configurable, step-by-step process. While we illustrate it here for a synthetic dataset like *Super-CLEVR*, the procedure is analogous for a real-world dataset like *ImageNet*, where "generation" is simulated via controlled subsampling of the original dataset.

1. *Define Rare Slice Candidates*: We first create a slice configuration file that specifies the set $S = \{c_i \mid c_i \text{ appears in a pair of } P\}$ of visually similar subclasses intended to be rare. For example, in *Super-CLEVR*, c_i can be the "dirtbike" subclass, which is paired with "mountain bike" in P because they are visually similar. In the same file, each subclass c_i may be restricted by specifying any combination of attribute values that makes the respective slice more specific, such as by fixing a particular colour and material. For example, a rare slice can be defined as the "dirtbike" subclass with colour "red" and material "metal". These user-specified attributes are exhaustively combined with all values of the remaining attributes. For example, if the attribute "size" is not specified, then the rare slice "dirtbike-red-metal" will include all possible values of "size", i.e. "dirtbike-red-metal-small" and "dirtbike-red-metal-large". Non-rare slices consist of all remaining combinations of subclasses and attribute values that are not rare slices. For example, the combination "dirtbike-blue-metal-small" is a non-rare slice because the user has restricted the rare slice to the colour "red". In summary, all $c_i \in S$, together with their user-specified attribute values, are defined as rare slices, while every other combination of subclass and attribute values is a non-rare slice.
2. *Set Occurrence Probability*: We assign to each rare slice a low occurrence probability α; lower α means a lower probability of creating an object.
3. *Configure Data Splits*: A second configuration file defines the total number n_s of objects per split $s \in \{train, validation\}$ and whether the split should contain rare slices. We typically generate the training split with rare slices and the validation split with a uniform distribution of all objects for fair evaluation.
4. *Rare and Non-Rare Slice Configuration*: A third configuration file is generated containing the complete specification of all rare and non-rare slices as defined in step 1 according to the class hierarchy $\mathcal{H}$.
5. *Generate or Subsample*: The configuration files are used to either guide the modified *Super-CLEVR* image generator or the *ImageNet* subsampling script to produce the final data splits. In both cases, the goal is to create data splits with the desired object distributions. In *Super-CLEVR*, the generator

computes the target number of rare objects for each subclass c_i in a split s as $n_{c_i,s} = \alpha \cdot n_s$, rounded to an integer. If $n_{c_i,s} = 0$, the configuration is invalid and the process stops with an error that no rare objects can be generated with the given n_s, asking the user to increase it. Otherwise, $n_{c_i,s}$ ($\geqslant 1$) rare objects are randomly distributed among the rendered images, with the remaining slots filled by $n_s - \sum_{i=1}^{n} n_{c_i,s}$ uniformly distributed non-rare objects.

Following the above steps, specific rare slices can be generated depending on the taxonomy under consideration. Furthermore, for each generated image, the *Super-CLEVR* generator produces the corresponding description consisting of the objects in the scene, their attributes, and the relationships between them. These descriptions allow scene graphs to be readily derived and then encoded into ILP examples, as shown in Fig. 3. For *ImageNet*, which provides a single class label per image without detailed object annotations, the necessary scene graphs are generated using external tools, as detailed in Section 6.

6. Experiments

The proposed SDM approach was evaluated in a series of experiments, which aimed to assess the effectiveness of rare slice generation, rule extraction, and model mending. To demonstrate the versatility of our approach, we performed the evaluation on two distinct benchmarks: the synthetic *Super-CLEVR* dataset for an object detection task, and the real-world *ImageNet* dataset for an image classification task. This section describes the evaluation platform, outlines the experimental setup for both benchmarks, and presents the results from our analysis. All data and details are available in the online repository.

6.1. Evaluation Platform

The evaluation platform is a server running Ubuntu 22.04.2 LTS (kernel version 6.8) with two Intel Xeon Silver 4314 CPUs (each having 16 cores at 2.40GHz, 2 threads per core, and 24MB of cache), 1,024GB of DRAM, four NVIDIA RTX A5000 GPUs (each having 24GB of VRAM), and the CUDA 12.2 API.

6.2. Super-CLEVR Experiments

This section details the experimental setup and presents the results from our evaluation of the proposed SDM architecture for the object detection task on the generated *Super-CLEVR* dataset. Specifically, we built a challenging and imbalanced training set and used it to train several *YOLOv5* models for object detection, each based on a different set of target classes from our custom taxonomies. Afterwards, we iteratively evaluated, diagnosed, and improved these models on the validation set. We outline the data taxonomies, dataset composition, the neural network architecture, and the iterative process of slice discovery and model mending in our pipeline.

6.2.1. Experimental Setup

In the following, we describe the experimental setup for each module of our SDM architecture.

Taxonomies. In our experiment, we used two custom taxonomies based on vehicle subclasses available in *Super-CLEVR*: airliner, biplane, fighter jet, private jet, sedan, minivan, station wagon, pickup truck, SUV, school bus, articulated bus, double bus, transit bus, scooter, chopper, sportbike, dirtbike, tandem bike, utility bike, mountain bike, and road bike. First, we identified five pairs of vehicle subclasses as visually similar: (“dirtbike”, “mountain bike”), (“articulated bus”, “transit bus”), (“utility bike”, “mountain bike”), (“pickup truck”, “sedan”), and (“private jet”, “airliner”). Then, we defined two *Super-CLEVR* taxonomies according to the proposed heuristic presented in Section 5, separating the vehicle subclasses of the pairs into different target classes. Specifically, as we descend toward the bottom of a taxonomy, more pairs are separated into distinct classes. In this way, we induced the generation of five rare slices to test the SDM implementation. To investigate rare slice generation, we defined from these taxonomies a total of five sets of target classes, referred to as *hierarchies*, each serving as training data labels to train a separate *YOLOv5* model. The two taxonomies are listed and described below:

1. *Vehicle Type (VT)* classifies vehicles according to their type in a multilevel taxonomy, where target classes become more and more specific at each level. The name of a class suggests the vehicles it contains. For example, the "air vehicle" class contains air vehicles such as "airliner" and "biplane". In contrast, the "scooter" and "mountain bike" vehicles are in the "land vehicle" class, but also in the class below, called "two-wheeler", since they only have two wheels. However, "scooter" belongs to the more specific "motorcycle" class while "mountain bike" is in the "bicycle" class. The same applies to the other classes and vehicle subclasses, as shown in Fig. 7, where vehicle subclasses are the leaves of the taxonomy. In the following, the four different hierarchies considered in the experiments to evaluate and compare the influence of rare slices on classification performance are reported. For each hierarchy of the *VT* taxonomy, the corresponding classes constitute the targets for training a neural network model.
   - *Hierarchy 1 (VT:$\mathcal{H}1$):* "air vehicle" and "land vehicle".
   - *Hierarchy 2 (VT:$\mathcal{H}2$):* "air vehicle", "two-wheeler", and "multi-wheeler".
   - *Hierarchy 3 (VT:$\mathcal{H}3$):* "air vehicle", "bicycle", "motorcycle", "bus", and "car". These are the classes that constitute the original hierarchy used in *Super-CLEVR*.
   - *Hierarchy 4 (VT:$\mathcal{H}4$):* "air vehicle", "sports bicycle", "urban bicycle", "sports motorcycle", "urban motorcycle", "regular bus", "specialized bus", "offroad car", and "urban car".

   Note that we purposely designed *VT:$\mathcal{H}1$* and *VT:$\mathcal{H}2$* not to satisfy the heuristic criterion, i.e. no previously defined pair of vehicle subclasses was separated into different target classes, serving as base cases. In contrast, for *VT:$\mathcal{H}3$* and *VT:$\mathcal{H}4$*, we defined the classes based on the heuristic criterion. In particular, *VT:$\mathcal{H}3$* only separates the ("dirtbike", "mountain bike") pair, and *VT-H4* separates four pairs: ("dirtbike", "mountain bike"), ("articulated bus", "transit bus"), ("utility bike", "mountain bike"), and ("pickup truck", "sedan"). This design allowed us to assess the effectiveness of the proposed heuristic in generating rare slices in the *YOLOv5* models trained for each hierarchy.
2. *Primary Purpose (PP)* classifies vehicles according to their primary use, as shown in Fig. 8. For example, "scooter" is in the "urban vehicle" class, which contains vehicles intended for urban transportation, while "dirtbike" is in the "offroad vehicle" class. For the *PP* taxonomy, we considered only one hierarchy, referred to as *PP:$\mathcal{H}1$*, with five target classes – "urban vehicle", "offroad vehicle", "specialized vehicle", "high-speed vehicle", and "recreational vehicle" – designed to separate four of the visually similar pairs of vehicle subclasses: ("articulated bus", "transit bus"), ("utility bike", "mountain bike"), ("pickup truck", "sedan"), and ("private jet", "airliner").

Dataset. For the two taxonomies, we generated a single training set of 10,000 images using the *Super-CLEVR* generator. Each image contains between three and six vehicles from the vehicle subclasses listed in Table 1. Vehicle attributes taken into account in image generation include: "materials" (e.g. "metal"), "colours" (e.g. "gray"), "sizes" (e.g. "small"), and "directions" (e.g. "southwest"). To create rare slices, we introduced data imbalance in the training set by manipulating the occurrence probability α of specific vehicle subclasses, without restricting them by specifying any combination of attribute values. Specifically, we selected one vehicle subclass from each pair mentioned above – "dirtbike", "articulated bus", "utility bike", "pickup truck", and "private jet" – as potential rare slice by setting its occurrence probability α to 0.05% of the total number n_s of vehicles in the training set. Depending on the hierarchy used in neural network training, these vehicle subclasses are potential rare slices; the remaining vehicle subclasses were uniformly distributed. Last, to fairly evaluate model performance, we generated a single validation set of 2,500 images with a balanced distribution, where each of the 21 vehicle subclasses is uniformly represented.

Neural Network. For each of the five hierarchies (VT:$\mathcal{H}1$ – VT:$\mathcal{H}4$ and PP:$\mathcal{H}1$), a *YOLOv5* model version *yolov5s*[2] was built on the training set running 80, 160, and 320 epochs using an image size of 640×640 pixels

[2]The *yolov5s* is the second-smallest pretrained version in the *YOLOv5* family. It trades off some accuracy for a much smaller size and faster inference.

```
hard(V0) :- contains(V0,V1), south(V1), utility(V1).
hard(V0) :- contains(V0,V1), north(V1), utility(V1).
hard(V0) :- contains(V0,V2), contains(V0,V1), small(V1), metal(V1), southeast(V2), utility(V2).
```

Fig. 4. Rules extracted by *Popper* for the "urban bicycle" class in $VT{:}\mathcal{H}4$, for sample size 100%. The rules correctly detect the rare slices "utility bike facing north" and "utility bike facing south".

and a batch size of 16. The default *YOLOv5* hyperparameters were used, including the *SGD* optimiser, initial learning rate of 0.01, final learning rate factor of 0.01, momentum of 0.937, and weight decay of 5.0×10^{-4}. Then, each trained model was evaluated on the validation set, and the results were inspected.

Rule Extraction and Selection. For the rule extraction module, we employ *Popper*, *FOLD-R++*, and *FastLAS* to identify rare slices within underperforming target classes of each hierarchy. The ILP systems extract rules based on the scene graphs generated for each image; an example is shown in Fig. 4. These rules consist of a combination of vehicle attributes, described earlier in Section 3.1. The process begins by identifying problematic target classes with *recall* at or below a predefined target class threshold τ_c. The value for τ_c is empirically determined by analysing the per-class model performance on the validation set. By placing the threshold within an observed performance gap, a criterion is created to separate underperforming classes that require intervention from those that are well-performing. For each problematic class, the ILP systems generate a set of rules describing rare slices that the model struggles to classify correctly. Then, we analyse these extracted rules to find unifying patterns and simplify them into more general candidate hypotheses derived by selecting the vehicle attributes that occurred more often in the extracted rules. For a candidate rule to be considered formally as a description of a potential rare slice, we introduce a rare slice hypothesis threshold τ_h. This threshold is also empirically determined and serves as a criterion to ensure that a candidate rule for a rare slice is supported by a significant percentage of the extracted rules. Its purpose is to filter out spurious or overly specific rules that might only be supported by a small fraction of the ILP hypotheses. The rationale behind this strategy is supported by the fact that if most of the extracted rules agree on the choice of a vehicle attribute, it means that such an attribute is more likely to be the most appropriate to characterise positive examples, i.e. rare slices. To ensure the robustness of our findings, we test each ILP system across various hyperparameter settings with a timeout of 3,600 seconds:

1. For *Popper*, we used the *noisy* mode, which allows it to learn the minimal description length program from noisy data. Furthermore, we varied the sample size of the validation set, using 25%, 50%, and 100%, to study the scalability as the amount of available data changed.
2. For *FOLD-R++*, we tested nine configurations by combining the three sample sizes (25%, 50%, 100%) with three different exception ratios (0.25, 0.50, 0.75). This hyperparameter represents the threshold of the ratio of false-positive examples (i.e. exceptions) to true-positive examples that a rule can entail.
3. For *FastLAS*, we used the *opl* mode, which runs the original *FastLAS1* algorithm. Furthermore, to use it in the mode that supports noisy data, we assigned a penalty to each example, which is the cost of violating it. As there are much less positive than negative examples and positive examples are more important to cover because they characterise rare slices, we set the penalty values for positive and negative examples to 4 and 2, respectively. To narrow down the hypothesis space, the maximum number of variables per rule was limited to 2. We also tested nine configurations for *FastLAS*, combining the three sample sizes with three rule head penalties (10, 20, 30) for the scoring function that charges such penalty values for each extracted rule head, to observe how they affect the quality of the output result.

All hyperparameter values mentioned were empirically fine-tuned by exploratory experimentation. Specifically, we selected several reasonable values to test different configurations of ILP systems in extracting meaningful rules describing rare slices. All other system hyperparameters use default values. A set of rules was obtained for each experimental configuration based on the hierarchy, target class, ILP system, and hyperparameter values considered. This comprehensive evaluation allows us to assess the effectiveness,

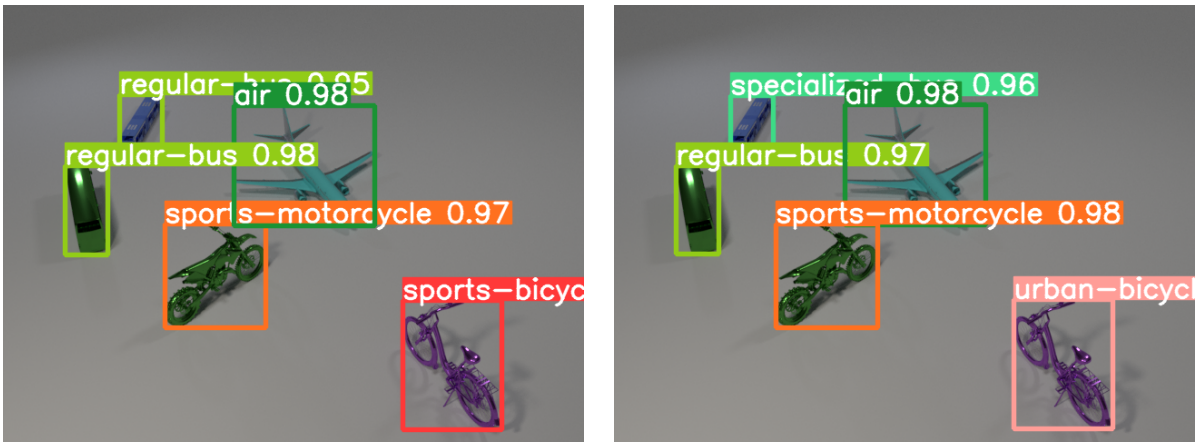

Fig. 5. The left figure shows a scene, based on *VT:H4*, in which vehicles corresponding to the "utility bike" and "articulated bus" rare slices are misclassified by *YOLOv5* into the "sports bicycle" and "regular bus" classes, respectively. In contrast, the right figure shows the same scene in which such vehicles are correctly classified, after model mending, into their "urban bicycle" and "specialized bus" classes, respectively.

speed, and verbosity of each ILP system and to verify that the identified rare slices are consistent across different configurations.

Model Mending. We proceed with the model mending step after selecting candidate rules that describe rare slices. We augment the original training set with new images to address data imbalance and further train the model. These images, generated with the *Super-CLEVR* data generator, specifically contain the vehicle attributes based on the rare slices identified by the selected rules. As for the neural network hyperparameters, the initial learning rate is modified according to the specific needs of each model mending iteration, as we will discuss below. All other neural network hyperparameters remain as in the initial model training (see Section 6.2.1). The validation set remains unchanged for all iterations of our SDM pipeline to ensure a fair and consistent evaluation of performance improvements.

6.2.2. Experimental Results

In the following, we present the experimental results for rare slice generation, rule extraction, and model mending from the iterative application of our SDM architecture.

Rare Slice Generation and Initial Model Training. Our method successfully generates rare slices on which *YOLOv5* models underperform, as revealed by the experimental results described below. More precisely, confusion matrices from model validation confirm the effectiveness of our taxonomy-based approach in inducing the presence of rare slices within the neural network models trained on hierarchies satisfying the proposed heuristic. Furthermore, rare slices degraded model performance across all epoch values considered (i.e. 80, 160, and 320), highlighting the persistence of rare slices even as the number of training rounds increases. The *YOLOv5* training process saves the model weights that achieve the highest performance on the validation set. Hence, 160-epoch models that performed best were designated as our baseline defective models. In our object detection task, model performance is measured by its *recall* metric, also known as *true positive rate (TPR)*, on the target classes of the validation set. Recall represents the proportion of all true positives that were correctly classified as positive. To diagnose the models, we inspected the recall of each target class. To identify underperforming classes, we set the target class threshold τ_c to 95.00%. Any target class performing at or below this threshold is considered problematic and is inspected via our SDM. As expected, models trained on *VT:H1* and *VT:H2*, which did not employ the subclass separation heuristic, showed no evidence of rare slices, achieving 100% recall for all target classes and thus exceeding τ_c, as shown in Fig. 10 and Fig. 13. In contrast, applying this heuristic to the hierarchy design consistently

resulted in models exhibiting potential rare slices, a finding substantiated by their confusion matrix recall values:

- For *VT:$\mathcal{H}3$*, the "motorcycle" class recall dropped to 94.00%, as shown in Fig. 16, falling below our performance bar τ_c and thus being marked as problematic. The other target classes in this hierarchy – "air vehicle", "bicycle", "bus", and "car" – all achieved 100.00% recall.
- *VT:$\mathcal{H}4$* was the most challenging case. Four of its target classes fell significantly below the threshold τ_c, as shown in Fig. 22: "urban bicycle" at 80.00%, "sports motorcycle" at 86.00%, "specialized bus" at 91.00%, and "offroad car" at 92.00%. These were all identified as problematic classes for rule extraction. In contrast, the other target classes performed well: the "regular bus" class achieved 99.00% recall, while the "air vehicle", "sports bicycle", "urban motorcycle", and "urban car" classes all reached 100.00% recall.
- For *PP:$\mathcal{H}1$*, four target classes performed at or below the threshold τ_c, as shown in Fig. 31, making them targets for diagnosis: "high-speed vehicle" at 92.00%, "specialized vehicle" at 95.00%, "urban vehicle" at 95.00%, and "offroad vehicle" at 95.00%. In contrast, the "recreational vehicle" class achieved 100.00% recall.

First Rule Extraction and Selection Iteration. The poor performance of the nine target classes suggests investigating them in search of rare slices. To this end, we employed our rule extraction module, tasking ILP systems to find the rules that identify rare slices in each problematic class of the models. After extracting the rules, we analysed them to identify underlying patterns. As previously mentioned, these rules consist of a combination of vehicle attributes. Our analysis revealed that the vehicle subclass was the primary feature in the extracted rules, often appearing with specific secondary vehicle attributes that further defined the potential rare slice. Therefore, we simplified all these rules into more general candidate hypotheses, such as "an image is difficult for the object detection model if it contains a dirtbike facing north". To formalise which of these candidate rules to consider as descriptions of potential rare slices, we set the rare slice hypothesis threshold τ_h to 33.33%. Consequently, only candidate rules that agree with a percentage of extracted rules greater than or equal to τ_h are retained. *Popper*, *FOLD-R++*, and *FastLAS* results, summarised in Tables 2-4, revealed patterns across all nine problematic classes. In the tables, an entry is marked with a ✓ to denote that at least one extracted rule agrees with a candidate rule for that target class, while an ✗ indicates that no extracted rule does. Furthermore, T denotes that the ILP system has timed out. Each mark is accompanied by a tuple where the first value is the runtime in seconds, the second is the total number of extracted rules, the third is the number of those rules that agree with the first candidate rule, the fourth is the number of those rules that agree with the second candidate rule, and so on. We now provide a comparison of the results in these tables between the ILP systems in terms of effectiveness, speed, and verbosity. Overall, *FOLD-R++* was the most effective and robust system. It successfully identified candidate rules for all nine problematic classes across the three hierarchies and for most hyperparameter configurations. In contrast, *FastLAS* demonstrated high potential but was less consistent. While it successfully identified several rare slices, it was prone to timeouts on larger sample sizes (e.g., for the "urban bicycle" and "high-speed vehicle" classes) and was highly sensitive to its rule head penalty hyperparameter. It was also generally the slowest system. *Popper* was the fastest and least verbose system, typically producing a small set of rules when successful. However, it was also the least effective, failing to identify three of the nine slices ("offroad car" in *VT:$\mathcal{H}4$*, and "offroad vehicle" and "specialized vehicle" in *PP:$\mathcal{H}1$*) and often requiring larger sample sizes to succeed on others. Despite these individual differences, the combined evidence from all three ILP systems strongly pointed towards the same underlying vehicle subclasses and their respective attributes, giving us high confidence in the subsequent hypothesis. A breakdown of the rules extracted for each underperforming class is given below:

- *VT:$\mathcal{H}3$*: For the "motorcycle" class, we found that 97.30% of rules involved the "dirtbike" subclass, and 89.19% also specified the "north" direction (Table 5). Since both these percentages exceed τ_h, this led to the selection of the more specific candidate rule for model mending:

Table 2
Rule extraction results of the first iteration of the *Popper* system on the models for *Super-CLEVR*.

	Sample Size		
	25%	50%	100%
VT:$\mathcal{H}3$			
M	✗ (2.94, 0, 0)	✓ (13.29, 1, 1)	✓ (29.77, 1, 1)
VT:$\mathcal{H}4$			
UB	✓ (11.34, 2, 1, 1)	✓ (29.44, 3, 1, 1)	✓ (71.48, 3, 1, 1)
SM	✓ (7.26, 1, 1)	✓ (26.19, 2, 1)	✓ (49.67, 1, 1)
OC	✗ (1.72, 0, 0)	✗ (7.09, 0, 0)	✗ (19.15, 0, 0)
SB	✗ (2.96, 0, 0)	✓ (8.80, 1, 1)	✓ (29.39, 2, 2)
PP:$\mathcal{H}1$			
UV	✓ (11.22, 2, 1, 1)	✓ (25.98, 2, 1, 1)	✓ (56.64, 3, 1, 1)
OV	✗ (3.58, 0, 0)	✗ (7.97, 0, 0)	✗ (30.55, 0, 0)
SV	✗ (1.95, 0, 0)	✗ (6.13, 0, 0)	✗ (17.74, 0, 0)
HV	✗ (7.22, 0, 0)	✗ (17.63, 0, 0)	✓ (46.47, 3, 3)

Table 3
Rule extraction results of the first iteration of the *FOLD-R++* system on the models for *Super-CLEVR*.

	Sample Size								
	25%			50%			100%		
	Exception ratio								
	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
VT:$\mathcal{H}3$									
M	✓ (37.69, 1, 1)	✓ (36.18, 1, 1)	✓ (37.26, 1, 1)	✓ (79.66, 1, 1)	✓ (77.82, 1, 1)	✓ (77.93, 1, 1)	✓ (290.86, 2, 1)	✓ (289.14, 2, 1)	✓ (182.39, 2, 1)
VT:$\mathcal{H}4$									
UB	✓ (35.28, 2, 1, 1)	✓ (37.01, 2, 1, 1)	✓ (37.22, 2, 1, 1)	✓ (234.91, 7, 1, 1)	✓ (236.99, 7, 1, 1)	✓ (122.71, 4, 1, 1)	✓ (649.15, 11, 1, 1)	✓ (577.53, 10, 1, 1)	✓ (454.25, 4, 1, 1)
SM	✗ (9.78, 0, 0)	✗ (9.78, 0, 0)	✗ (8.19, 0, 0)	✓ (154.98, 3, 1)	✓ (204.66, 3, 1)	✓ (204.32, 3, 1)	✓ (795.97, 5, 3)	✓ (521.79, 3, 1)	✓ (526.68, 3, 1)
OC	✓ (58.85, 1, 1)	✓ (59.03, 1, 1)	✓ (58.62, 1, 1)	✗ (25.89, 0, 0)	✗ (25.80, 0, 0)	✗ (25.81, 0, 0)	✓ (806.72, 7, 7)	✓ (814.38, 7, 7)	✓ (811.59, 7, 7)
SB	✓ (89.33, 2, 1)	✓ (163.07, 4, 2)	✓ (123.45, 3, 1)	✓ (103.69, 1, 1)	✓ (104.96, 1, 1)	✓ (184.92, 2, 1)	✓ (2,020.21, 12, 9)	✓ (798.17, 4, 2)	✓ (737.88, 4, 1)
PP:$\mathcal{H}1$									
UV	✓ (136.68, 5, 2, 1)	✓ (131.18, 3, 1, 1)	✓ (131.81, 3, 1, 1)	✓ (366.44, 5, 2, 1)	✓ (275.22, 3, 1, 1)	✓ (269.19, 3, 1, 1)	✓ (2,376.56, 15, 1, 6)	✓ (562.03, 3, 1, 1)	✓ (564.16, 3, 1, 1)
OV	✓ (107.50, 4, 4)	✓ (106.64, 4, 4)	✓ (106.28, 4, 4)	✓ (268.38, 4, 4)	✓ (270.10, 4, 4)	✓ (269.29, 4, 4)	✓ (260.71, 1, 1)	✓ (261.95, 1, 1)	✓ (260.29, 1, 1)
SV	✓ (154.18, 3, 2)	✓ (152.29, 3, 2)	✓ (152.81, 3, 2)	✗ (26.51, 0, 0)	✗ (26.45, 0, 0)	✗ (25.57, 0, 0)	✓ (1,957.50, 6, 5)	✓ (1,954.41, 6, 5)	✓ (1,949.11, 6, 5)
HV	✓ (269.40, 5, 5)	✓ (267.56, 5, 5)	✓ (115.80, 2, 2)	✓ (448.65, 4, 4)	✓ (441.62, 4, 4)	✓ (442.57, 4, 4)	✓ (722.85, 3, 3)	✓ (602.35, 3, 3)	✓ (563.96, 3, 3)

Table 4
Rule extraction results of the first iteration of the *FastLAS* system on the models for *Super-CLEVR*.

	Sample Size								
	25%			50%			100%		
	Rule head penalty								
	10	20	30	10	20	30	10	20	30
VT:$\mathcal{H}3$									
M	✓ (164.45, 1, 1)	✗ (168.09, 0, 0)	✗ (170.47, 0, 0)	✓ (710.42, 6, 6)	✓ (703.62, 3, 3)	✗ (709.12, 0, 0)	✓ (1,979.61, 7, 6)	✓ (2,000.79, 4, 4)	✓ (1,979.56, 2, 2)
VT:$\mathcal{H}4$									
UB	✓ (428.70, 1, 1, 1)	✓ (429.37, 1, 1, 1)	✓ (432.62, 1, 1, 1)	✓ (1,777.84, 10, 8, 7)	✓ (1,732.70, 4, 2, 3)	✓ (1,747.49, 2, 2, 1)	*T*	*T*	*T*
SM	✓ (267.49, 5, 4)	✓ (278.79, 1, 1)	✓ (276.20, 1, 1)	✓ (1,242.71, 12, 8)	✓ (1,232.06, 4, 3)	✓ (1,187.63, 1, 1)	✓ (3,185.80, 11, 8)	✓ (3,145.65, 6, 6)	✓ (3,117.27, 1, 1)
OC	✗ (130.18, 1, 0)	✗ (134.03, 0, 0)	✗ (135.46, 0, 0)	✓ (387.37, 3, 2)	✗ (390.18, 0, 0)	✗ (397.25, 0, 0)	✓ (1,237.95, 4, 1)	✗ (1,243.66, 0, 0)	✗ (1,219.60, 0, 0)
SB	✓ (183.55, 2, 1)	✗ (183.69, 0, 0)	✗ (175.57, 0 ,0)	✓ (506.58, 4, 1)	✗ (513.43, 0, 0)	✗ (513.36, 0, 0)	✓ (1,804.56, 5, 3)	✓ (1,759.75, 2, 2)	✗ (1,787.29, 0, 0)
PP:$\mathcal{H}1$									
UV	✓ (587.16, 5, 3, 4)	✓ (580.60, 2, 2, 2)	✓ (574.57, 1, 1, 1)	✓ (1,902.39, 8, 6, 3)	✓ (1,930.73, 3, 3, 2)	✓ (1,939.26, 2, 2, 1)	*T*	*T*	*T*
OV	✗ (187.68, 0, 0)	✗ (185.68, 0, 0)	✗ (190.05, 0, 0)	✓ (722.63, 3, 2)	✗ (721.56, 0, 0)	✗ (712.37, 0, 0)	✓ (2,587.70, 5, 2)	✓ (2,571.52, 2, 1)	✗ (2,584.88, 0, 0)
SV	✗ (132.89, 1, 0)	✗ (138.17, 0, 0)	✗ (134.52, 0, 0)	✓ (402.53, 1, 1)	✗ (415.38, 0, 0)	✗ (402.71, 0, 0)	✗ (1,550.34, 1, 0)	✗ (1,557.28, 0, 0)	✗ (1,538.56, 0, 0)
HV	✓ (415.23, 4, 3)	✓ (422.52, 1, 1)	✗ (413.03, 0, 0)	✓ (1,193.53, 5, 3)	✓ (1,176.64, 1, 1)	✗ (1,189.77, 0, 0)	*T*	*T*	*T*

Table 5
Rule extraction results of the "motorcycle" class of the first iteration on the *VT:$\mathcal{H}3$* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	46.00	2	**Dirtbike**: **2** (**100.00**%) $\left[\textbf{North}: \textbf{2}\ (\textbf{100.00}\%)\right]$
FOLD-R++	1,108.93	12	**Dirtbike**: **12** (**100.00**%) $\left[\textbf{North}: \textbf{9}\ (\textbf{75.00}\%)\right]$
FastLAS	8,586.13	23	**Dirtbike**: **22** (**95.65**%) $\left[\textbf{North}: \textbf{22}\ (\textbf{95.65}\%)\right]$, Sedan: 1 (4.35%)
Total	9,741.06	37	**Dirtbike**: **36** (**97.30**%) $\left[\textbf{North}: \textbf{33}\ (\textbf{89.19}\%)\right]$, Sedan: 1 (2.70%)

Table 6
Rule extraction results of the "specialized bus" class of the first iteration on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	41.15	3	**Articulated Bus**: **3** (**100.00**%) $\left[\textbf{North}: \textbf{3}\ (\textbf{100.00}\%)\right]$
FOLD-R++	4,325.68	33	**Articulated Bus**: **33** (**100.00**%) $\left[\textbf{North}: \textbf{19}\ (\textbf{57.58}\%)\right]$, Tandem Bike: 1 (3.03%), Utility Bike: 1 (3.03%)
FastLAS	7,427.78	13	**Articulated Bus**: **11** (**84.62**%) $\left[\textbf{North}: \textbf{7}\ (\textbf{53.85}\%)\right]$, Pickup Truck: 1 (7.69%), Biplane: 1 (7.69%), Dirtbike: 1 (7.69%), Road Bike: 1 (7.69%), Fighter Jet: 1 (7.69%)
Total	11,794.61	49	**Articulated Bus**: **47** (**95.92**%) $\left[\textbf{North}: \textbf{29}\ (\textbf{59.18}\%)\right]$, Pickup Truck: 1 (2.04%), Biplane: 1 (2.04%), Dirtbike: 1 (2.04%), Road Bike: 1 (2.04%), Fighter Jet: 1 (2.04%), Tandem Bike: 1 (2.04%), Utility Bike: 1 (2.04%)

Table 7
Rule extraction results of the "high-speed vehicle" class of the first iteration on the *PP:$\mathcal{H}1$* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	71.32	3	**Private Jet**: **3** (**100.00**%) $\left[\textbf{Large} - \textbf{Metal}: \textbf{3}\ (\textbf{100.00}\%)\right]$, Chopper: 1 (33.33%)
FOLD-R++	3,874.76	33	**Private Jet**: **33** (**100.00**%) $\left[\textbf{Large} - \textbf{Metal}: \textbf{33}\ (\textbf{100.00}\%)\right]$, Sportbike: 5 (15.15%), School Bus: 5 (15.15%)
FastLAS	15,610.72	11	**Private Jet**: **10** (**90.91**%) $\left[\textbf{Large} - \textbf{Metal}: \textbf{8}\ (\textbf{72.73}\%)\right]$, Sedan: 1 (9.09%)
Total	19,556.80	47	**Private Jet**: **46** (**97.87**%) $\left[\textbf{Large} - \textbf{Metal}: \textbf{44}\ (\textbf{93.62}\%)\right]$, Sportbike: 5 (10.64%), School Bus: 5 (10.64%), Sedan: 1 (2.13%), Chopper: 1 (2.13%)

- Motorcycle *first* (and only)[3] candidate rule:
  `hard(V0) :- contains(V0,V1), dirtbike(V1), north(V1).`

where `V1` denotes a vehicle in an image `V0`.

– *VT:$\mathcal{H}4$*: For the four problematic classes, the percentage of rules identifying the rare slice exceeded the threshold τ_h in all cases.

- For "specialized bus", 95.92% of rules identified "articulated bus", with 59.18% specifying "north" direction (Table 6).
- For "offroad car", 84.38% of rules identified "pickup truck" and "rubber" material (Table 18).
- For "sports motorcycle", 86.36% of rules identified "dirtbike", with 66.67% also specifying "north" direction (Table 19).
- For "urban bicycle", 98.68% of rules identified "utility bike", with directions "north" (35.53%) and "south" (34.21%) being the most common secondary attributes (Table 20).

This led to the selection of the following candidate rules for model mending:

- Specialized bus *first* candidate rule:
  `hard(V0) :- contains(V0,V1), articulated_bus(V1), north(V1).`
- Offroad car *first* candidate rule:
  `hard(V0) :- contains(V0,V1), pickup_truck(V1), rubber(V1).`
- Sports motorcycle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), dirtbike(V1), north(V1).`
- Urban bicycle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), utility_bike(V1), north(V1).`
- Urban bicycle *second* candidate rule:
  `hard(V0) :- contains(V0,V1), utility_bike(V1), south(V1).`

where `V1` denotes a vehicle in an image `V0`.

– *PP:$\mathcal{H}1$*: The ILP systems also found strong evidence for rare slices in this hierarchy, with all candidate hypotheses surpassing the threshold τ_h.

- For "high-speed vehicle", 97.87% of rules identified "private jet", with 93.62% specifying "large" and "metal" attributes (Table 7).
- For "offroad vehicle", 89.19% of rules identified "pickup truck", with 86.49% also specifying "rubber" material (Table 21).
- For "specialized vehicle", 96.67% of rules identified "articulated bus", with 73.33% specifying "north" direction (Table 22).
- For "urban vehicle", 95.77% of rules identified "utility bike", frequently with "north" (43.66%) and "south" (42.25%) directions (Table 23).

This led to the selection of the following candidate rules for model mending:

- High-speed vehicle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), private_jet(V1), large(V1), metal(V1).`
- Offroad vehicle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), pickup_truck(V1), rubber(V1).`
- Specialized vehicle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), articulated_bus(V1), north(V1).`
- Urban vehicle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), utility_bike(V1), north(V1).`
- Urban vehicle *second* candidate rule:
  `hard(V0) :- contains(V0,V1), utility_bike(V1), south(V1).`

where `V1` denotes a vehicle in an image `V0`.

[3]This clarification also applies to several subsequent candidate rules and will not be repeated, to avoid redundancy.

First Model Mending Iteration. To address the data imbalance without introducing catastrophic forgetting[4], we augmented the original training set with new images generated by the *Super-CLEVR* generator according to the selected rules. This augmentation was done according to the rules selected in the previous step, which consist of both the rare vehicle subclass and its most common secondary attributes. For each hierarchy, the respective defective model (the best performing 160-epoch version) was then retrained on its newly balanced dataset for 20, 40, and 80 epochs, using the same hyperparameters as before. By comparing the outcomes of the three numbers of retraining epochs for each model, we determined the most effective model mending for each specific hierarchy. The results from the number of optimal retraining epochs are detailed below.

- For *VT:$\mathcal{H}3$*, the original training set was augmented with 500 new images of the "dirtbike" rare slice adhering to its secondary vehicle attribute of facing "north". The most effective model retraining was the 20-epoch one, which successfully addressed the deficiency in the "motorcycle" class by increasing its recall from 94.00% to 99.00%, as shown in Fig. 17. The remaining classes – "air vehicle", "bicycle", "bus", and "car" – all maintained their 100.00% recall.
- For *VT:$\mathcal{H}4$*, the training set was augmented with 500 new images for each of the four identified rare slices, adhering to their secondary vehicle attributes: "utility bike" facing "north" or "south", "dirtbike" facing "north", "articulated bus" facing "north", and "pickup truck" made of "rubber". The 20-epoch retraining was the most effective and led to substantial improvements. The recall for the "urban bicycle" class rose from 80.00% to 94.00%, "sports motorcycle" from 86.00% to 96.00%, "specialized bus" from 91.00% to 96.00%, and "offroad car" from 92.00% to 98.00%, as shown in Fig. 23. The latter three classes became well-detected, and the "urban bicycle" class was significantly improved. However, "urban bicycle" is the only class to fall below our target class threshold τ_c of 95.00%, marking it as the target for a second SDM iteration. Finally, the target classes that already performed well were not negatively impacted; "regular bus" recall improved from 99.00% to 100.00%, while "air vehicle" maintained its 100.00% recall. The other classes saw only a negligible 1.00% drop in recall, indicating that the mending process did not cause significant catastrophic forgetting.
- For *PP:$\mathcal{H}1$*, the training set was augmented with 500 new images for each of its four rare slices defined by their primary and secondary vehicle attributes: "utility bike" facing "north" or "south", "articulated bus" facing "north", "pickup truck" made of "rubber", and "private jet" in both "large" and "metal". The most effective model retraining was the 40-epoch one, which achieved notable performance gains. The recall for the "urban vehicle" and "offroad vehicle" classes both rose from 95.00% to 97.00%. Similarly, the "specialized vehicle" recall increased from 95.00% to 98.00%, and "high-speed vehicle" from 92.00% to 96.00%, as shown in Fig. 32. Finally, the already well-performing "recreational vehicle" class maintained its 100.00% recall.

The mending process was successful across all hierarchies, substantially improving the recall of the target classes; notably, the overall performance of the model for VT:$\mathcal{H}4$ improved substantially. The three previously underperforming classes "sports motorcycle", "specialized bus", and "offroad car" now reported recalls meeting the target class threshold τ_c of 95.00%, indicating that their initial rare slices had been successfully resolved. However, we conducted a second iteration of the SDM pipeline to investigate any remaining deficiencies.

Second Rule Extraction and Selection Iteration for VT:$\mathcal{H}4$. Despite a significant improvement from 80.00% to 94.00%, the "urban bicycle" class was the only one that still failed to satisfy our threshold τ_c. We again employed our rule extraction module, tasking ILP systems to find rules that identify a potential rare slice within this problematic class. As in the previous iteration, we analysed the rules to identify underlying patterns and used the same hypothesis formation process and rare slice hypothesis threshold τ_h of 33.33%. The results of this second rule extraction iteration for the "urban bicycle" class are

[4] *Catastrophic forgetting* [20], also known as *catastrophic interference*, refers to the tendency of a neural network to rapidly and drastically forget previously learned information upon learning new information.

Table 8
Rule extraction results of the second iteration of the *Popper* system on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*.

	Sample Size		
	25%	50%	100%
VT:$\mathcal{H}4$			
UB	✗ (1.38, 0, 0)	✗ (4.33, 0, 0)	✓ (11.37, 1, 1)

Table 9
Rule extraction results of the second iteration of the *FOLD-R++* system on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*.

	Sample Size								
	25%			50%			100%		
	Exception ratio								
	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
VT:$\mathcal{H}4$									
UB	✓ (37.07, 5, 5)	✓ (44.94, 8, 8)	✓ (22.09, 4, 4)	✓ (217.11, 3, 3)	✓ (215.44, 3, 3)	✓ (216.16, 4, 4)	✓ (500.27, 10, 5)	✓ (188.49, 2, 2)	✓ (170.81, 7, 7)

Table 10
Rule extraction results of the second iteration of the *FastLAS* system on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*.

	Sample Size								
	25%			50%			100%		
	Rule head penalty								
	10	20	30	10	20	30	10	20	30
VT:$\mathcal{H}4$									
UB	✗ (67.49, 0, 0)	✗ (66.89, 0, 0)	✗ (68.64, 0, 0)	✗ (218.81, 0, 0)	✗ (219.82, 0, 0)	✗ (217.62, 0, 0)	✓ (582.65, 5, 2)	✗ (589.29, 0, 0)	✗ (584.73, 0, 0)

Table 11
Rule extraction results of the "urban bicycle" class of the second iteration on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	17.08	1	**Utility Bike**: **1** (**100.00**%) [**North**: **1** (**100.00**%)]
FOLD-R++	1,612.38	46	**Utility Bike**: **43** (**93.48**%) [**North**: **41** (**89.13**%)], Tandem Bike: 7 (15.22%), Sedan: 2 (4.35%), Scooter: 2 (4.35%), Sportbike: 1 (2.17%), Private Jet: 1 (2.17%), Station Wagon: 1 (2.17%)
FastLAS	2,615.94	5	**Utility Bike**: **2** (**40.00**%) [**North**: **2** (**40.00**%)], Road Bike: 1 (20.00%), Pickup Truck: 1 (20.00%), School Bus: 1 (20.00%)
Total	4,245.40	52	**Utility Bike**: **46** (**88.46**%) [**North**: **44** (**84.62**%)], Tandem Bike: 7 (13.46%), Sedan: 2 (3.85%), Scooter: 2 (3.85%), Sportbike: 1 (1.92%), Private Jet: 1 (1.92%), Station Wagon: 1 (1.92%), Road Bike: 1 (1.92%), Pickup Truck: 1 (1.92%), School Bus: 1 (1.92%)

summarised in Tables 8-10. The ILP systems showed a greater divergence in performance in this iteration. *FOLD-R++* was once again the most effective system. It successfully found candidate rules across all nine hyperparameter configurations. It was also the most verbose, generating a total of 46 rules, 41 of which agreed with our candidate hypothesis across all configuration settings. In contrast, *Popper* and *FastLAS* were far less effective. *Popper* only succeeded in one configuration (at 100.00% sample size) and was the least verbose, generating only a single rule. *FastLAS* also only succeeded in one configuration (at 100.00% sample size with a rule head penalty of 10) and failed in all others, generating just 5 rules in total. This

suggests that the remaining performance issue was more subtle to identify. A detailed breakdown of the rules extracted for the underperforming "urban bicycle" class is provided in Table 11. The evidence pointed to the "utility bike" subclass as the primary source of the problem. In particular, 88.46% of all extracted rules involved the "utility bike" subclass, and 84.62% also specified the "north" direction. We hypothesised that this problem resulted from the difficulty of the model in distinguishing the "utility bike" vehicle, when facing "north", from other visually similar subclasses (e.g., "mountain bike"), a confusion that was not completely resolved by the initial data augmentation. Since both these percentages exceed τ_h, this led to the selection of the more specific candidate rule for the second model mending:

– Urban bicycle *first* candidate rule:
`hard(V0) :- contains(V0,V1), utility_bike(V1), north(V1).`

where, as before, `V1` denotes a vehicle in an image `V0`.

Second Model Mending Iteration for VT:$\mathcal{H}4$. For the second mending iteration, we augmented the training data by generating 500 new images with the *Super-CLEVR* generator for the "utility bike" rare slice adhering to its secondary vehicle attribute of facing "north". We then retrained the best-performing model from the first iteration (the one mended over 20 epochs) for an additional 20, 40, and 80 epochs. To refine the model without risking degrading its performance for classes that still rely heavily on previous training, we employed fine-tuning with a lower initial learning rate of 0.001.

The model retrained for an additional 20 epochs proved to be the most effective. This second intervention successfully resolved the persistent slice, as shown in Fig. 24. The recall for the problematic "urban bicycle" class rose significantly from 94.00% to 98.00%, finally surpassing our target class threshold τ_c. The other classes maintained their high performance, with some showing minor fluctuations representing an acceptable trade-off for the significant improvement in the underperforming class: "sports motorcycle" recall remained at 96.00%, while "specialized bus" saw a slight single-point increase to 97.00%. The "offroad car" class saw a negligible decrease from 98.00% to 97.00%, and the "sports bicycle" class also saw a slight decrease from 99.00% to 98.00%. The "urban motorcycle" class improved from 99.00% to 100.00% recall, "air vehicle" and "regular bus" maintained their 100.00% recall, and "urban car" remained stable at 99.00%. This confirms that the iterative mending process was highly successful in correcting a specific deficiency without causing significant degradation elsewhere. Therefore, the process was terminated at this stage.

6.3. ImageNet Experiments

This section details the experimental setup and presents the results from our evaluation of the proposed SDM architecture for the image classification task on a curated subset of the *ImageNet* dataset. Specifically, we built a challenging and imbalanced training set and used it to train a *YOLOv5* model for image classification. Afterwards, we iteratively evaluated, diagnosed, and improved such a model on the respective validation set. As for *Super-CLEVR*, we outline the data taxonomy, dataset composition, the neural network architecture, and the iterative process of slice discovery and model mending in our pipeline.

6.3.1. Experimental Setup

In the following, we describe the experimental setup for each module of our SDM architecture.

Taxonomy. In our experiment, we used a reduced list of 11 vehicle subclasses from *ImageNet*: tandem bicycle, motorhome, moped, scooter, mountain bike, jeep, pickup truck, station wagon, convertible, minivan, and moving van. First, we identified the following four pairs of vehicle subclasses as visually similar: ("tandem bicycle", "mountain bike"), ("moped", "mountain bike"), ("jeep", "minivan"), and ("station wagon", "minivan"). Then, as for *Super-CLEVR*, we defined a taxonomy according to the proposed heuristic presented in Section 5, separating the vehicle subclasses of the pairs into different target classes. This taxonomy, which we refer to as the *Vehicle (VE)* taxonomy, classifies vehicles according to their type as illustrated in Fig. 9. For example, the "scooter" subclass is in the "motorcycle" class, while the "pickup truck" subclass is in the "offroad vehicle" class. To investigate rare slice generation, we defined from the *VE* taxonomy a single set of target classes, referred to as *Hierarchy 1 (VE:$\mathcal{H}1$)*, serving as training data labels

to train a *YOLOv5* model. *VE:$\mathcal{H}$1* comprises the following classes: "leisure vehicle" (LV), "motorcycle" (M), "offroad vehicle" (OV), "passenger car" (PC), and "van" (V). We specifically structured these classes to create challenging classification scenarios by separating all previous pairs of vehicle subclasses. In this way, we induced the generation of rare slices to test the SDM implementation.

Dataset. We built our training and validation sets using a subset of *ImageNet*. The training set consists of 3,634 images distributed across the five classes of our *VE* taxonomy. To create rare slices, we intentionally varied the number of images per vehicle subclass, introducing data imbalance. Specifically, we designated four vehicle subclasses – "tandem bicycle", "moped", "jeep", and "station wagon" – as rare slices. These subclasses were chosen from each of the four visually similar pairs mentioned above. The occurrence probability α for each of these subclasses in the training set was set to 5% of the respective target class, thus making them potential rare slices. For this training set, rare slices were defined without considering specific values for vehicle attributes, such as colour or position. All remaining subclasses were uniformly distributed, with each represented by 500 images. To fairly evaluate model performance, we created a separate validation set of 2,200 images with a balanced distribution, where each of the 11 vehicle subclasses is uniformly represented. For the ILP systems, we generated scene graphs for each image with the help of the *GPT-4.1*[5] VLM and then manually curated the results to ensure accuracy. This assisted annotation process allowed us to capture key attributes for both the scene environment and the vehicles. Environment attributes include "number of persons" (e.g. 2), "background" (e.g. "rural outdoor"), "snow" (e.g. "false"), and "time of day" (e.g. "daytime"). Vehicle attributes include "colour" (e.g. "black"), "orientation" (e.g. "side view"), "position" (e.g. "foreground"), "type" (e.g. "tandem bicycle"), and "visibility" (e.g. "fully visible").

Neural Network. For the *VE:$\mathcal{H}$1* hierarchy, a *YOLOv5* model version *yolov5s-cls*[6] was built on the training set running 20, 40, and 80 epochs using an image size of 224×224 pixels and a batch size of 16. The default *YOLOv5* hyperparameters were used, including the *Adam* optimiser, initial learning rate of 0.001, final learning rate factor of 0.01, momentum of 0.9, and weight decay of 5.0×10^{-5}. Then, each trained model was evaluated on the validation set, and the results were inspected.

Rule Extraction and Selection. For the rule extraction module, we employ the same ILP systems (*Popper*, *FOLD-R++*, and *FastLAS*) and a similar methodology as described in the *Super-CLEVR* experiments to identify rare slices within underperforming target classes. The process again begins by identifying problematic target classes, for which the ILP systems then extract rules based on the scene graphs generated for each image. These rules consist of a combination of vehicle and environment attributes, described in Section 6.3.1. The differences in this setup are as follows:

- Problematic target classes are those with a *Top-1 accuracy* at or below a predefined target class threshold τ_c.
- While the hyperparameter settings for *Popper* and *FOLD-R++* remain the same as in *Super-CLEVR*, for *FastLAS* we tested nine configurations combining the three sample sizes (25%, 50%, 100%) with three different rule head penalty values (1, 5, 10). These values were empirically fine-tuned based on exploratory experimentation specifically for the *ImageNet* dataset. We observed during initial runs that the higher penalty values used in the *Super-CLEVR* experiments were too restrictive in the *ImageNet* context, often preventing *FastLAS* from learning any rules at all.

Then, the extracted rules are analysed to form candidate hypotheses, which are formally selected if they meet a predefined rare slice hypothesis threshold τ_h. As before, this comprehensive evaluation measures the effectiveness, speed, and verbosity of each ILP system, while also verifying the consistency of the identified rare slices.

[5] We used version *gpt-4.1-2025-04-14* by OpenAI [44].

[6] The *yolov5s-cls* is the second-smallest pretrained version in the *YOLOv5* family. It trades off some accuracy for a much smaller size and faster inference.

Fig. 6. The left figure shows a scene, based on *VE:$\mathcal{H}$1*, in which the vehicle corresponding to the "tandem bicycle" rare slice is misclassified by *YOLOv5* into the "offroad vehicle" class. In contrast, the right figure shows the same scene in which such a vehicle is correctly classified, after model mending, into its "leisure vehicle" class.

Model Mending. We proceed with the model mending step using the same procedure as in the *Super-CLEVR* experiments. After selecting candidate rules that describe rare slices, we augment the original training set with new images to address the data imbalance and then further train the model to mend its behaviour. The primary difference in this setup is the source of the new data. Instead of using a generator, the new images are sourced from the *ImageNet* dataset, specifically chosen based on the rare slices identified by the selected rules. As before, the initial learning rate is modified according to the specific needs of each model mending iteration, while all other neural network hyperparameters remain the same as those used in the initial model training and described in Section 6.3.1.

6.3.2. Experimental Results

In the following, we present the experimental results for rare slice generation, rule extraction, and model mending from the iterative application of our SDM architecture.

Rare Slice Generation and Initial Model Training. In our image classification task, model performance is measured by its *Top-1 accuracy*, which represents the percentage of validation images where the main prediction of the model matches the correct label. The *YOLOv5* training process saves the model weights that achieve the highest Top-1 accuracy on the validation set. We trained the model for 20, 40, and 80 epochs; the results are shown in Table 12. The model trained for 40 epochs yielded the best initial result, achieving an overall Top-1 accuracy of 80.32% on the validation set, compared to 79.23% for 20 epochs and 77.73% for 80 epochs, respectively. This model was designated as our baseline defective model. To diagnose the model, we inspected the Top-1 accuracy of each of the five target classes, as shown in Table 12. To identify underperforming classes, we set the target class threshold τ_c to 86.00%. Any target class performing at or below this threshold is considered problematic and is inspected via our SDM. Our analysis confirmed that four target classes fell below this performance bar: "leisure vehicle" at 62.25%, "motorcycle" at 83.50%, "offroad vehicle" at 85.00%, and "passenger car" at 80.75%. In contrast, the "van" class exceeded the threshold with an accuracy of 88.00%.

First Rule Extraction and Selection Iteration. The poor performance of the four target classes suggests investigating them in search of rare slices. To this end, we employed our rule extraction module, tasking ILP systems to find the rules that identify rare slices in each problematic class of the model. After extracting the rules, we analysed them to identify underlying patterns. As previously mentioned, these rules consist of a combination of vehicle and environment attributes. However, the vehicle subclass was the unifying feature in most rules for each potential rare slice. Therefore, we simplified these observations into more general candidate hypotheses, such as "an image is difficult for the model to classify if it represents a tandem

Table 12

Top-1 accuracy of the *VE:$\mathcal{H}$1* model on the *ImageNet* validation set after the initial model training. The left table shows the overall accuracy at different training epochs, the right table the accuracy for each of the five target classes.

Epochs	Top-1 acc. (%)
20	79.23
40	**80.32**
80	77.73

Target class	Top-1 acc. (%)
LV	**62.25**
M	**83.50**
OV	**85.00**
PC	**80.75**
V	88.00

bicycle". To formalise which of these candidate rules to consider as descriptions of potential rare slices, we set the rare slice hypothesis threshold τ_h to 33.33%. Consequently, only candidate rules that agree with a percentage of extracted rules greater than or equal to τ_h are retained. The results for *Popper*, *FOLD-R++*, and *FastLAS*, summarised in Tables 13-15, revealed patterns across all four problematic classes. We now compare the results in these tables for the ILP systems in terms of effectiveness, speed, and verbosity.

FastLAS was the most effective and robust system, successfully identifying candidate rules for all four problematic classes across almost all hyperparameter settings. However, it was also the slowest and by far the most verbose, generating the most "noisy" output. *FastLAS* produced a total of 408 rules across the four classes, many of which were non-contributing rules. In contrast, *FOLD-R++* was extremely fast, moderately verbose, but slightly less effective, finding candidate rules for three of the four problematic classes while generating a total of 90 rules. It completely missed the "offroad vehicle" class. *Popper* performed the worst, identifying only the candidate rule for the "leisure vehicle" class and failing on the other three. It was also the least verbose system, generating a mere 3 rules in total. Despite these individual differences, the combined evidence from all three ILP systems strongly pointed towards the same underlying vehicle subclasses, giving us high confidence in the subsequent hypothesis. A detailed breakdown of the rules extracted for each underperforming class is provided in Table 16 and Appendix D. In particular, for the "leisure vehicle" class, we found that 72.97% of the rules involved the "tandem bicycle" subclass. Similarly, "moped" was present in 79.61% of the rules for the "motorcycle" class, "jeep" in 61.00% for the "offroad vehicle" class, and "station wagon" in 74.34% for the "passenger car" class. Since each of these percentages exceeds τ_h, this led to the selection of the following candidate rules for model mending:

- Leisure vehicle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), tandem_bicycle(V1).`
- Motorcycle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), moped(V1).`
- Offroad vehicle *first* candidate rule:
  `hard(V0) :- contains(V0,V1), jeep(V1).`
- Passenger car *first* candidate rule:
  `hard(V0) :- contains(V0,V1), station_wagon(V1).`

where `V1` denotes a vehicle in an image `V0`.

First Model Mending Iteration. To address the data imbalance without introducing catastrophic forgetting, we augmented the original training set with new images taken from *ImageNet* according to the selected rules. This augmentation aimed to precisely balance the distribution of all vehicle subclasses to a target of 500 images each. The number of new images added for each vehicle subclass was therefore the exact amount needed to reach this target from their initial count in the imbalanced set. Specifically, we added 473 new images for "tandem bicycle", 473 for "moped", 447 for "jeep", and 473 for "station wagon". The defective model (the 40-epoch version) was then retrained on this newly balanced dataset for 10 and 20 epochs, using

Table 13
Rule extraction results of the first iteration of the *Popper* system on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Sample size		
	25%	50%	100%
VE:$\mathcal{H}$1			
LV	✓ (0.95, 1, 1)	✓ (3.12, 1, 1)	✓ (5.49, 1, 1)
M	✗ (0.75, 0, 0)	✗ (2.24, 0, 0)	✗ (2.53, 0, 0)
OV	✗ (1.25, 0, 0)	✗ (2.73, 0, 0)	✗ (4.62, 0, 0)
PC	✗ (0.65, 0, 0)	✗ (2.40, 0, 0)	✗ (3.96, 0, 0)

Table 14
Rule extraction results of the first iteration of the *FOLD-R++* system on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Sample size								
	25%			50%			100%		
	Exception ratio								
	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
VE:$\mathcal{H}$1									
LV	✗ (0.01, 1, 0)	✗ (0.01, 1, 0)	✗ (0.01, 1, 0)	✓ (0.06, 10, 10)	✓ (0.01, 4, 1)	✓ (0.01, 4, 1)	✓ (0.07, 8, 8)	✓ (0.01, 1, 1)	✓ (0.01, 1, 1)
M	✓ (0.01, 5, 5)	✓ (0.02, 8, 6)	✓ (0.02, 8, 6)	✓ (0.02, 3, 3)	✓ (0.01, 2, 2)	✓ (0.01, 1, 1)	✓ (0.02, 2, 2)	✓ (0.02, 2, 2)	✓ (0.02, 2, 2)
OV	✗ (0.01, 0, 0)	✗ (0.01, 0, 0)	✗ (0.01, 0, 0)	✗ (0.02, 1, 0)	✗ (0.02, 1, 0)	✗ (0.02, 1, 0)	✗ (0.02, 1, 0)	✗ (0.02, 1, 0)	✗ (0.02, 1, 0)
PC	✓ (0.01, 3, 1)	✓ (0.01, 3, 1)	✓ (0.01, 3, 1)	✓ (0.02, 4, 4)	✓ (0.01, 2, 2)	✓ (0.01, 2, 2)	✓ (0.02, 1, 1)	✓ (0.02, 1, 1)	✓ (0.02, 1, 1)

Table 15
Rule extraction results of the first iteration of the *FastLAS* system on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Sample size								
	25%			50%			100%		
	Rule head penalty								
	1	5	10	1	5	10	1	5	10
VE:$\mathcal{H}$1									
LV	✓ (2.30, 20, 16)	✓ (2.11, 17, 15)	✓ (2.06, 5, 3)	✓ (8.39, 21, 16)	✓ (8.17, 18, 14)	✓ (8.27, 8, 6)	✓ (25.42, 29, 18)	✓ (23.60, 25, 16)	✓ (24.12, 8, 6)
M	✓ (2.01, 7, 5)	✓ (1.82, 7, 5)	✓ (1.73, 1, 1)	✓ (6.90, 11, 7)	✓ (6.92, 11, 8)	✓ (6.74, 1, 1)	✓ (13.74, 15, 12)	✓ (12.79, 14, 11)	✓ (13.08, 3, 3)
OV	✓ (3.94, 9, 7)	✓ (3.63, 9, 6)	✓ (3.51, 1, 1)	✓ (8.77, 15, 10)	✓ (8.11, 13, 9)	✓ (8.25, 1, 1)	✓ (26.98, 25, 15)	✓ (26.10, 20, 12)	✗ (25.88, 1, 0)
PC	✓ (2.49, 10, 6)	✓ (2.17, 10, 7)	✗ (2.16, 0, 0)	✓ (7.12, 14, 11)	✓ (6.44, 14, 11)	✓ (6.30, 2, 2)	✓ (13.96, 22, 17)	✓ (14.25, 19, 14)	✓ (14.11, 2, 2)

the same hyperparameters as before. As shown in Table 17, the model retrained for just 10 epochs achieved the highest Top-1 accuracy of 90.55%, while 20 epochs yielded a slightly lower accuracy of 90.09%. The intervention was highly effective, marking a significant improvement over the baseline. A detailed look at the per-class performance for the best model (retrained for 10 epochs), shown in Table 17 and exemplified in Fig. 6, confirms this. The Top-1 accuracies for the four problematic classes rose substantially: "leisure vehicle" at 90.00%, "motorcycle" at 96.00%, "offroad vehicle" at 90.50%, and "passenger car" at 90.25%. In contrast, the "van" class deteriorated from 88.00% to 85.75%, probably due to the improvement in the accuracy of visually similar vehicles with which it is confused. This made "van" the new lowest-performing class and the only one to fall below our target class threshold τ_c of 86.00%, marking it as the target for a second SDM iteration. The details of this second iteration of our SDM pipeline are described in Appendix D.

Table 16
Rule extraction results of the "leisure vehicle" class of the first iteration on the *VE:$\mathcal{H}1$* model for *ImageNet*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	9.56	3	**Tandem Bicycle**: **3** (**100.00**%)
FOLD-R++	0.20	31	**Tandem Bicycle**: **22** (**70.97**%)
FastLAS	104.44	151	**Tandem Bicycle**: **110** (**72.85**%), Motorhome: 13 (8.61%), Mountain Bike: 6 (3.97%), Pickup Truck: 2 (1.32%), Station Wagon: 1 (0.66%)
Total	114.20	185	**Tandem Bicycle**: **135** (**72.97**%), Motorhome: 13 (7.03%), Mountain Bike: 6 (3.24%), Pickup Truck: 2 (1.08%), Station Wagon: 1 (0.54%)

Table 17
Top-1 accuracy of the *VE:$\mathcal{H}1$* model on the *ImageNet* validation set after the first model mending iteration. The left table shows the overall accuracy at different training epochs, the right table the accuracy for each of the five target classes.

Epochs	Top-1 acc. (%)
10	**90.55**
20	90.09

Target class	Top-1 acc. (%)
LV	90.00
M	96.00
OV	90.50
PC	90.25
V	**85.75**

7. Discussion

Our experiments, conducted on both the synthetic *Super-CLEVR* and real-world *ImageNet* datasets, demonstrate that the proposed SDM is highly effective at identifying rare slices in CV models. By systematically training, diagnosing, and mending models for both object detection (for *Super-CLEVR*) and image classification (for *ImageNet*) tasks, we validated the general efficacy of our neurosymbolic approach. The taxonomy-based heuristic at the core of our approach consistently and successfully induced challenging, hard-to-detect rare slices in the trained models. This allowed for a rigorous evaluation of the SDM pipeline. The subsequent application of ILP systems not only identified underperforming slices, but also extracted interpretable logical rules that pinpointed the specific data attributes causing the model to underperform. These rules then guided a targeted data augmentation and model mending process, which led to significant and consistent performance improvements across all tested hierarchies. In the sequel, we compare the performance of the integrated ILP systems, discuss the impact of model mending, and briefly compare our work with some existing SDMs. Finally, we acknowledge current limitations.

7.1. Comparison of ILP Systems

Our comparative analysis among the three ILP systems – *Popper*, *FOLD-R++*, and *FastLAS*– reveals the differences in their performance:

- *Popper* was the fastest and least verbose system, but also the least effective. It failed to identify several key rare slices, and its success was highly dependent on having a large sample size of data.
- *FOLD-R++* was a very reliable and robust system with respect to hyperparameters. It successfully identified the underlying rare slices in nearly all problematic classes across both *Super-CLEVR* and *ImageNet* experiments, even with smaller data samples. The *exception ratio* in *FOLD-R++* had a minimal impact on its rule extraction, indicating limited sensitivity to this hyperparameter in our context.

– *FastLAS* was the most effective and expressive system, capable of identifying subtle rare slices, as demonstrated in the second iteration of the *ImageNet* experiments where other systems failed. However, this expressiveness comes at a cost. *FastLAS* was consistently the slowest system, and highly sensitive to its *rule head penalty* hyperparameter. Lower penalty values consistently produced meaningful rules, whereas higher values sometimes prevented the system from discovering any rules.

In addition to requiring less data for slice discovery, smaller samples have the advantage of significantly reducing the running time of ILP systems. In particular, using smaller samples was necessary for *FastLAS* in the *Super-CLEVR* experiments. One possible reason why *FastLAS* is slower may be its penalty mechanism and scoring function, which make the optimisation problem more challenging to solve. On the other hand, one probable explanation for the speed of *Popper* is the lack of negation in its extracted rules. Indeed, it is important to consider that both *FOLD-R++* and *FastLAS* provide rules with negation, which greatly widens the hypothesis space. However, rules with negation allow for alternative and more concise descriptions of slices by specifying which vehicle attributes should not be present. The ability to express rules with negation may be one reason why *FOLD-R++* and *FastLAS* have succeeded more than *Popper* in identifying rare slices. Despite their individual differences, the ILP systems showed a crucial consistency; when multiple systems succeeded, they invariably pointed to the same root cause (e.g., a specific vehicle subclass), reinforcing the validity of our findings. This convergence gives us high confidence in the identified slices and the subsequent mending strategies. Interestingly, while larger sample sizes generally improved the likelihood of success for all systems, both *FOLD-R++* and *FastLAS* were often effective with as little as 25.00% of the validation data, highlighting the potential for efficient application in resource-constrained scenarios. Finally, these findings emphasise the importance of achieving a good trade-off between speed, effectiveness, and robustness in ILP-based slice discovery.

7.2. Impact of Model Mending

The model mending phase, guided by the rules extracted via our SDM, proved highly effective across all hierarchies. By augmenting the training set with new images specifically targeting the identified rare slices, we achieved substantial improvements in model performance. For instance, in the challenging VT:$\mathcal{H}4$ hierarchy, the recall for the "urban bicycle" class jumped from 80.00% to 94.00% after the first mending iteration. The iterative nature of our SDM pipeline proved to be very effective. The second mending iteration for VT:$\mathcal{H}4$ further improved the "urban bicycle" recall to 98.00%, demonstrating the capacity of our SDM to solve progressively more subtle performance issues. This iterative refinement also highlights its ability to enhance model robustness without causing catastrophic forgetting, as the performance of already well-behaving classes remained high.

YOLOv5 models achieved high overall performance on the *Super-CLEVR* dataset, with mAP@0.5 values approaching 1.0 in all experiments. However, the goal of this work was not to maximise general object detection metrics, but rather to diagnose and correct highly specific, induced failures known as rare slices. For this purpose, per-class recall serves as a more precise diagnostic tool than a global metric like mAP. While a high mAP score confirms the good overall model performance, it can mask the poor performance on a specific, underrepresented slice of data, as the error is averaged out. By focusing on the recall of the problematic classes, we can directly measure the impact of the slice and, more importantly, verify the success of the mending process in a targeted manner. While the main analysis focuses on recall to clearly illustrate the diagnosis and repair of rare slices, a more comprehensive set of performance metrics is provided for completeness. We have included detailed results in Appendix C., which contains the confusion matrices, F1-Confidence curves, and other model training and validation performance metrics (e.g., mAP@0.5) for all *Super-CLEVR* hierarchies, both before and after model mending. This supplementary data confirms that the targeted improvements in recall discussed in the main text are accompanied by corresponding positive gains in the F1-score, reinforcing the overall efficacy of the proposed SDM pipeline.

Finally, the results across both the *Super-CLEVR* and *ImageNet* experiments indicate that model mending allows for significant improvements without extensive retraining. For the *Super-CLEVR* hierarchies, a

relatively small number of additional training epochs – between 20 and 40 – was sufficient to integrate the new data and correct the identified rare slices. This efficiency was even more pronounced in the *ImageNet* experiments, where a brief retraining of just 10 epochs yielded effective results for both mending iterations.

7.3. Comparison with Existing Methods

As detailed in Section 2, the state-of-the-art can be broadly categorised, and our neurosymbolic architecture offers a distinct alternative that emphasises logical rule extraction. Several methods in slice discovery and rare data mining, such as *Domino* [17], *Spotlight* [13], *George* [55], and TALISMAN [30], have introduced strategies to identify rare or underperforming data regions by operating largely in embedding spaces or latent distributions. Another relevant method by Jiang et al. [24] proposed density-based *rare example mining* using normalizing flows over learned detection features in a 3D object detection setting. Although this approach significantly improves performance on rare intraclass instances, it does not provide semantic explanations of errors or insight into the nature of failure modes. In fact, the common limitation of these approaches is a lack of interpretability. A rare slice is typically identified as a cluster of data points, not a human-understandable concept in a semantic, logical format. More recent methods, such as *PromptAttack* [40], *AdaVision* [21], and *SSD-LLM* [38], leverage the power of large-scale generative and multimodal models. These approaches can explore and structure datasets to suggest potential areas of underperformance. However, the final rare slice description often remains a textual prompt or a collection of images, rather than a formal, verifiable rule.

Our neurosymbolic SDM contrasts with these methods by prioritising interpretability and targeted causality. The main technical difference is the use of ILP to move from systematic model errors to a set of compact, human-readable, and formal logical rules describing them. This provides several advantages:

1. *Transparency*: Logical rules offer a clear semantic explanation of the failure condition of the model (e.g., `shape(utility)` and `direction(north)`).
2. *Editability*: Rules are not only descriptive, but also prescriptive. They provide a precise, actionable specification that can directly guide the model mending process.
3. *Debugging*: Rules serve as a valuable tool for model debugging, allowing ML practitioners to understand the specific visual attributes that confuse the model.

In summary, our work addresses the fundamental challenge of making slice discovery interpretable and directly actionable for targeted model correction.

7.4. Limitations

For each CV task, our SDM approach builds on the availability of scene graph representations to extract interpretable logical rules describing "hidden" rare slices, i.e. underperforming subsets of data not explicitly labelled and difficult to spot from unstructured data, such as images. These representations provide the rich semantic structure necessary for our method. However, scene graphs are generally not readily available for datasets, especially in real-world scenarios. Nevertheless, rapid advances in Vision Language Models (VLMs) are making it increasingly feasible to semi-automatically generate (curated generation) such semantic representations from image data. In our ongoing work, we are actively and systematically exploring the integration of VLMs to automate the scene graph generation step, as we have experimented here for real-world images from the *ImageNet* dataset. Consequently, our SDM is directly applicable whenever appropriate semantic representations can be obtained from image data, making automated semantic extraction a promising direction for our future work.

A second limitation is the current reliance on manual, exploratory tuning for the hyperparameters of the ILP systems. While our experiments show that robust rules can be found by testing a range of configurations, this process can be time-consuming and may require domain expertise.

Furthermore, the scalability of ILP systems can be computationally intensive, especially with large validation sets or with a complex hypothesis space defined by numerous attributes and predicates. As

observed in our experiments, *FastLAS* was prone to timeouts when analysing the entire validation set in *Super-CLEVR*, indicating that the ILP system performance can be a bottleneck.

Finally, the current implementation of our SDM focuses on discovering rare slices defined by object attributes (e.g., "a yellow rubber utility bike facing south"). A limitation of this approach is that it does not yet consider slices defined by the relationships between objects (e.g., "a bicycle next to a car"). Extending the proposed SDM to incorporate object relations would allow for the discovery of more complex rare slices, providing deeper insights into model failures.

8. Conclusion and Future Work

In this work, we have presented a neurosymbolic approach to address the slice discovery problem. In particular, we have provided a modular architecture and an implementation that connects dataset generation, model classification, and rule extraction via ILP to identify misclassified rare slices. Our experiments, conducted on both the synthetic *Super-CLEVR* and real-world *ImageNet* datasets, demonstrate the effectiveness of our methodology. The proposed taxonomy-based heuristic reliably generated datasets with predictable rare slices, validating our approach for inducing specific model failures. The ILP systems proved effective at producing useful logical rules describing rare slices. Further training the models with new data guided by these rules resulted in significant performance improvements on the problematic classes for both object detection and image classification tasks. Our SDM approach can also be extended to the multi-label classification task, thus dealing with taxonomies structured as directed acyclic graphs in addition to tree-shaped ones. Furthermore, our results underscore the effectiveness of the iterative nature of the SDM pipeline. We showed that further iteration can successfully diagnose and resolve more subtle and persistent errors, demonstrating the ability of the SDM to progressively refine model performance and address increasingly difficult deficiencies.

The results obtained are encouraging and demonstrate the potential of simultaneously exploiting DL and KRR methods for slice discovery. In this way, compact and human-readable logical rules can be extracted that improve the interpretability and explainability of a CV model under examination, also paving the way to advanced concepts such as causal and contrastive explanations.

Ongoing and Future Work. Although our experiments confirm that rule-based augmentation helps improve classification performance, we acknowledge that the overall effectiveness of model mending may vary depending on the specific rules extracted. A systematic analysis of the sensitivity of model mending to rule quality, granularity, and specificity remains as a promising issue for future work. Furthermore, to make the SDM pipeline more accessible and efficient, automated methods for setting the optimal hyperparameters for the various ILP systems could be explored, reducing the need for manual, exploratory tuning. Our *ImageNet* experiments relied on a VLM to generate the necessary scene graphs, as mentioned in Section 6.3.1. A key future direction is to systematically explore and integrate state-of-the-art VLMs to fully automate this step, rather than being provided with ground truth scene graphs, as is the case with *Super-CLEVR*. Creating a robust pipeline for generating high-quality semantic representations directly from input images will make our SDM framework scalable and applicable to any visual dataset. Using such tools for our SDM presents an interesting research challenge. Thus far, our work has focused on object attributes. We plan to extend the framework to incorporate relationships between objects. This will allow for the discovery of more specific rare slices (e.g., "a bicycle next to a car"), providing deeper insights into model failures at the expense of higher computational cost. Moreover, exploring the integration of further ILP systems, such as recent neurosymbolic ones, e.g. δILP [16] and αILP [53], as well as other rule learning approaches, such as those provided by Statistical Relational Learning, e.g. LERND [39], is on our agenda.

Acknowledgements

The project leading to this research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 101034440.

Additionally, this research was funded in whole or in part by the Austrian Science Fund (FWF) 10.55776/COE12, and it was supported by Bosch Center for AI (BCAI) in Renningen, Germany.

References

[1] V. Boreiko, M. Hein and J.H. Metzen, Identifying Systematic Errors in Object Detectors with the SCROD Pipeline, in: *Proc. IEEE/CVF International Conference on Computer Vision*, 2023, pp. 4090–4099.

[2] I. Bratko, *Prolog Programming for Artificial Intelligence, 4th Edition*, Addison-Wesley, 2012. ISBN 978-0-3214-1746-6.

[3] I. Bratko and S.H. Muggleton, Applications of Inductive Logic Programming, *Commun. ACM* **38**(11) (1995), 65–70. doi:10.1145/219717.219771.

[4] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D.M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever and D. Amodei, Language Models are Few-Shot Learners, in: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle, M. Ranzato, R. Hadsell, M. Balcan and H. Lin, eds, 2020. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

[5] J. Buolamwini and T. Gebru, Gender Shades: Intersectional Accuracy Disparities in Commercial Gender Classification, in: *Conference on Fairness, Accountability and Transparency, FAT 2018, 23-24 February 2018, New York, NY, USA*, S.A. Friedler and C. Wilson, eds, Proceedings of Machine Learning Research, Vol. 81, PMLR, 2018, pp. 77–91. http://proceedings.mlr.press/v81/buolamwini18a.html.

[6] Y. Chung, T. Kraska, N. Polyzotis, K.H. Tae and S.E. Whang, Slice Finder: Automated Data Slicing for Model Validation, in: *35th IEEE International Conference on Data Engineering, ICDE 2019, Macao, China, April 8-11, 2019*, IEEE, 2019, pp. 1550–1553. doi:10.1109/ICDE.2019.00139.

[7] M. Collevati, T. Eiter and N. Higuera, Leveraging Neurosymbolic AI for Slice Discovery, in: *Neural-Symbolic Learning and Reasoning - 18th International Conference, NeSy 2024, Barcelona, Spain, September 9-12, 2024, Proceedings, Part I*, T.R. Besold, A. d'Avila Garcez, E. Jiménez-Ruiz, R. Confalonieri, P. Madhyastha and B. Wagner, eds, LNCS, Vol. 14979, Springer, 2024, pp. 403–418. doi:10.1007/978-3-031-71167-1_22.

[8] B.O. Community, *Blender - a 3D modelling and rendering package*, Blender Foundation, Stichting Blender Foundation, Amsterdam, 2018. http://www.blender.org.

[9] A. Cropper and S. Dumancic, Inductive Logic Programming At 30: A New Introduction, *J. Artif. Intell. Res.* **74** (2022), 765–850. doi:10.1613/JAIR.1.13507. https://doi.org/10.1613/jair.1.13507.

[10] A. Cropper and R. Morel, Learning programs by learning from failures, *Mach. Learn.* **110**(4) (2021), 801–856. doi:10.1007/S10994-020-05934-Z. https://doi.org/10.1007/s10994-020-05934-z.

[11] A. Cropper, S. Dumancic, R. Evans and S.H. Muggleton, Inductive logic programming at 30, *Mach. Learn.* **111**(1) (2022), 147–172. doi:10.1007/S10994-021-06089-1. https://doi.org/10.1007/s10994-021-06089-1.

[12] W. Dai, Q. Xu, Y. Yu and Z. Zhou, Bridging Machine Learning and Logical Reasoning by Abductive Learning, in: *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, H.M. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E.B. Fox and R. Garnett, eds, 2019, pp. 2811–2822. https://proceedings.neurips.cc/paper/2019/hash/9c19a2aa1d84e04b0bd4bc888792bd1e-Abstract.html.

[13] G. d'Eon, J. d'Eon, J.R. Wright and K. Leyton-Brown, The Spotlight: A General Method for Discovering Systematic Errors in Deep Learning Models, in: *FAccT '22: 2022 ACM Conference on Fairness, Accountability, and Transparency, Seoul, Republic of Korea, June 21 - 24, 2022*, ACM, 2022, pp. 1962–1981. doi:10.1145/3531146.3533240.

[14] T. DeVries, I. Misra, C. Wang and L. van der Maaten, Does Object Recognition Work for Everyone?, in: *IEEE Conference on Computer Vision and Pattern Recognition Workshops, CVPR Workshops 2019, Long Beach, CA, USA, June 16-20, 2019*, Computer Vision Foundation / IEEE, 2019, pp. 52–59. http://openaccess.thecvf.com/content_CVPRW_2019/html/cv4gc/de_Vries_Does_Object_Recognition_Work_for_Everyone_CVPRW_2019_paper.html.

[15] D.P. Enot and R.D. King, Application of Inductive Logic Programming to Structure-Based Drug Design, in: *Knowledge Discovery in Databases: PKDD 2003, 7th European Conference on Principles and Practice of Knowledge Discovery in Databases, Cavtat-Dubrovnik, Croatia, September 22-26, 2003, Proceedings*, N. Lavrac, D. Gamberger, H. Blockeel and L. Todorovski, eds, LNCS, Vol. 2838, Springer, 2003, pp. 156–167. doi:10.1007/978-3-540-39804-2_16.

[16] R. Evans and E. Grefenstette, Learning Explanatory Rules from Noisy Data, *J. Artif. Intell. Res.* **61** (2018), 1–64. doi:10.1613/JAIR.5714. https://doi.org/10.1613/jair.5714.

[17] S. Eyuboglu, M. Varma, K.K. Saab, J. Delbrouck, C. Lee-Messer, J. Dunnmon, J. Zou and C. Ré, Domino: Discovering Systematic Errors with Cross Modal Embeddings, in: *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*, OpenReview.net, 2022. https://openreview.net/forum?id=FPCMqjI0jXN.

[18] P.W. Finn, S.H. Muggleton, D. Page and A. Srinivasan, Pharmacophore Discovery Using the Inductive Logic Programming System PROGOL, *Mach. Learn.* **30**(2–3) (1998), 241–270. doi:10.1023/A:1007460424845.

[19] C.D. Francescomarino, I. Donadello, C. Ghidini, F.M. Maggi, W. Rizzi and S. Tessaris, Making Sense of Temporal Event Data:A Framework for Comparing Techniques for the Discovery of Discriminative Temporal Patterns, in: *Advanced Information Systems Engineering - 36th International Conference, CAiSE 2024, Limassol, Cyprus, June 3-7, 2024, Proceedings*, G. Guizzardi, F.M. Santoro, H. Mouratidis and P. Soffer, eds, LNCS, Vol. 14663, Springer, 2024, pp. 423–439. doi:10.1007/978-3-031-61057-8_25.

[20] R.M. French, Catastrophic forgetting in connectionist networks, *Trends in Cognitive Sciences* **3**(4) (1999), 128–135. doi:https://doi.org/10.1016/S1364-6613(99)01294-2. https://www.sciencedirect.com/science/article/pii/S1364661399012942.

[21] I. Gao, G. Ilharco, S.M. Lundberg and M.T. Ribeiro, Adaptive Testing of Computer Vision Models, in: *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, IEEE, 2023, pp. 3980–3991. doi:10.1109/ICCV51070.2023.00370.

[22] M.A. Hedderich, J. Fischer, D. Klakow and J. Vreeken, Label-Descriptive Patterns and Their Application to Characterizing Classification Errors, in: *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, K. Chaudhuri, S. Jegelka, L. Song, C. Szepesvári, G. Niu and S. Sabato, eds, Proceedings of Machine Learning Research, Vol. 162, PMLR, 2022, pp. 8691–8707. https://proceedings.mlr.press/v162/hedderich22a.html.

[23] P. Hitzler and M.K. Sarker (eds), *Neuro-Symbolic Artificial Intelligence: The State of the Art*, Frontiers in Artificial Intelligence and Applications, Vol. 342, IOS Press, 2021. ISBN 978-1-64368-244-0. doi:10.3233/FAIA342.

[24] C.M. Jiang, M. Najibi, C.R. Qi, Y. Zhou and D. Anguelov, Improving the Intra-class Long-Tail in 3D Detection via Rare Example Mining, in: *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part X*, S. Avidan, G.J. Brostow, M. Cissé, G.M. Farinella and T. Hassner, eds, LNCS, Vol. 13670, Springer, 2022, pp. 158–175. doi:10.1007/978-3-031-20080-9_10.

[25] J. Johnson, B. Hariharan, L. van der Maaten, L. Fei-Fei, C.L. Zitnick and R.B. Girshick, CLEVR: A Diagnostic Dataset for Compositional Language and Elementary Visual Reasoning, in: *2017 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Honolulu, HI, USA, July 21-26, 2017*, IEEE Computer Society, 2017, pp. 1988–1997. doi:10.1109/CVPR.2017.215.

[26] N. Johnson, Á.A. Cabrera, G. Plumb and A. Talwalkar, Where Does My Model Underperform? A Human Evaluation of Slice Discovery Algorithms, in: *Proc. AAAI Conference on Human Computation and Crowdsourcing*, Vol. 11, 2023, pp. 65–76.

[27] S.N. Kalid, K. Khor, K.H. Ng and G. Tong, Detecting Frauds and Payment Defaults on Credit Card Data Inherited With Imbalanced Class Distribution and Overlapping Class Problems: A Systematic Review, *IEEE Access* **12** (2024), 23636–23652. doi:10.1109/ACCESS.2024.3362831.

[28] A. Koenecke, A. Nam, E. Lake, J. Nudell, M. Quartey, Z. Mengesha, C. Toups, J.R. Rickford, D. Jurafsky and S. Goel, Racial disparities in automated speech recognition, *Proc. Natl. Acad. Sci. USA* **117**(14) (2020), 7684–7689. doi:10.1073/PNAS.1915768117. https://doi.org/10.1073/pnas.1915768117.

[29] G. Kókai, Z. Alexin and T. Gyimóthy, Application of Inductive Logic Programming for Learning ECG Waveforms, in: *Artificial Intelligence Medicine, 6th Conference on Artificial Intelligence in Medicine in Europe, AIME'97, Grenoble, France, March 23-26, 1997, Proceedings*, E.T. Keravnou, C. Garbay, R.H. Baud and J.C. Wyatt, eds, LNCS, Vol. 1211, Springer, 1997, pp. 126–129. doi:10.1007/BFB0029443. https://doi.org/10.1007/BFb0029443.

[30] S. Kothawade, S. Ghosh, S. Shekhar, Y. Xiang and R.K. Iyer, Talisman: Targeted Active Learning for Object Detection with Rare Classes and Slices Using Submodular Mutual Information, in: *Computer Vision - ECCV 2022 - 17th European Conference, Tel Aviv, Israel, October 23-27, 2022, Proceedings, Part XXXVIII*, S. Avidan, G.J. Brostow, M. Cissé, G.M. Farinella and T. Hassner, eds, LNCS, Vol. 13698, Springer, 2022, pp. 1–16. doi:10.1007/978-3-031-19839-7_1.

[31] A. Krizhevsky, I. Sutskever and G.E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, *Commun. ACM* **60**(6) (2017), 84–90. doi:10.1145/3065386.

[32] N. Lavrac and S. Dzeroski, *Inductive Logic Programming - Techniques and Applications*, Ellis Horwood series in artificial intelligence, Ellis Horwood, 1994. ISBN 978-0-13-457870-5.

[33] M. Law, A. Russo and K. Broda, The ILASP system for learning Answer Set Programs, 2015. https://www.ilasp.com/.

[34] M. Law, A. Russo, E. Bertino, K. Broda and J. Lobo, FastLAS: Scalable Inductive Logic Programming Incorporating Domain-Specific Optimisation Criteria, in: *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, New York, NY, USA, February 7-12, 2020*, AAAI Press, 2020, pp. 2877–2885. doi:10.1609/AAAI.V34I03.5678. https://doi.org/10.1609/aaai.v34i03.5678.

[35] H. Li, G. Zhu, L. Zhang, Y. Jiang, Y. Dang, H. Hou, P. Shen, X. Zhao, S.A.A. Shah and M. Bennamoun, Scene Graph Generation: A comprehensive survey, *Neurocomputing* **566** (2024), 127052. doi:10.1016/J.NEUCOM.2023.127052. https://doi.org/10.1016/j.neucom.2023.127052.

[36] Z. Li, X. Wang, E. Stengel-Eskin, A. Kortylewski, W. Ma, B.V. Durme and A.L. Yuille, Super-CLEVR: A Virtual Benchmark to Diagnose Domain Robustness in Visual Reasoning, in: *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*, IEEE, 2023, pp. 14963–14973. doi:10.1109/CVPR52729.2023.01437.

[37] V. Lifschitz, *Answer Set Programming*, Springer, 2019. ISBN 978-3-030-24657-0. doi:10.1007/978-3-030-24658-7.

[38] Y. Luo, R. An, B. Zou, Y. Tang, J. Liu and S. Zhang, LLM as Dataset Analyst: Subpopulation Structure Discovery with Large Language Model, in: *Computer Vision - ECCV 2024 - 18th European Conference, Milan, Italy, September 29-October 4, 2024, Proceedings, Part XXXIII*, A. Leonardis, E. Ricci, S. Roth, O. Russakovsky, T. Sattler and G. Varol, eds, LNCS, Vol. 15091, Springer, 2024, pp. 235–252. doi:10.1007/978-3-031-73414-4_14.

[39] I. Merkys, crunchiness/lernd: LERND - implementation of ∂ILP, Zenodo, 2020. doi:10.5281/zenodo.4294059. https://github.com/crunchiness/lernd.

[40] J.H. Metzen, R. Hutmacher, N.G. Hua, V. Boreiko and D. Zhang, Identification of Systematic Errors of Image Classifiers on Rare Subgroups, in: *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, IEEE, 2023, pp. 5041–5050. doi:10.1109/ICCV51070.2023.00467.

[41] S.H. Muggleton, Inductive Logic Programming, *New Gener. Comput.* **8**(4) (1991), 295–318. doi:10.1007/BF03037089.

[42] L. Oakden-Rayner, J. Dunnmon, G. Carneiro and C. Ré, Hidden stratification causes clinically meaningful failures in machine learning for medical imaging, in: *ACM CHIL '20: ACM Conference on Health, Inference, and Learning, Toronto, Ontario, Canada, April 2-4, 2020 [delayed]*, M. Ghassemi, ed., ACM, 2020, pp. 151–159. doi:10.1145/3368555.3384468.

[43] V. Olesen, N. Weng, A. Feragen and E. Petersen, Slicing Through Bias: Explaining Performance Gaps in Medical Image Analysis Using Slice Discovery Methods, in: *Ethics and Fairness in Medical Imaging - Second International Workshop on Fairness of AI in Medical Imaging, FAIMI 2024, and Third International Workshop on Ethical and Philosophical Issues in Medical Imaging, EPIMI 2024, Held in Conjunction with MICCAI 2024, Marrakesh, Morocco, October 6-10, 2024, Proceedings*, E. Puyol-Antón, G. Zamzmi, A. Feragen, A.P. King, V. Cheplygina, M. Ganz-Benjaminsen, E. Ferrante, B. Glocker, E. Petersen, J.S.H. Baxter, I. Rekik and R. Eagleson, eds, LNCS, Vol. 15198, Springer, 2024, pp. 3–13. doi:10.1007/978-3-031-72787-0_1.

[44] OpenAI, Introducing GPT-4.1 in the API, 2025. https://openai.com/index/gpt-4-1/.

[45] T. Pedersen, S. Patwardhan and J. Michelizzi, WordNet: : Similarity - Measuring the Relatedness of Concepts, in: *Proc. Nineteenth National Conference on Artificial Intelligence, Sixteenth Conference on Innovative Applications of Artificial Intelligence, July 25-29, 2004, San Jose, California, USA*, D.L. McGuinness and G. Ferguson, eds, AAAI Press / The MIT Press, 2004, pp. 1024–1025. http://www.aaai.org/Library/AAAI/2004/aaai04-160.php.

[46] A. Radford, J.W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark, G. Krueger and I. Sutskever, Learning Transferable Visual Models From Natural Language Supervision, in: *Proc. 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, M. Meila and T. Zhang, eds, Proceedings of Machine Learning Research, Vol. 139, PMLR, 2021, pp. 8748–8763. http://proceedings.mlr.press/v139/radford21a.html.

[47] B. Recht, R. Roelofs, L. Schmidt and V. Shankar, Do ImageNet Classifiers Generalize to ImageNet?, in: *Proc. 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, K. Chaudhuri and R. Salakhutdinov, eds, Proceedings of Machine Learning Research, Vol. 97, PMLR, 2019, pp. 5389–5400. http://proceedings.mlr.press/v97/recht19a.html.

[48] J. Redmon, S.K. Divvala, R.B. Girshick and A. Farhadi, You Only Look Once: Unified, Real-Time Object Detection, in: *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, IEEE Computer Society, 2016, pp. 779–788. doi:10.1109/CVPR.2016.91.

[49] S. Sagadeeva and M. Boehm, SliceLine: Fast, Linear-Algebra-based Slice Finding for ML Model Debugging, in: *SIGMOD '21: International Conference on Management of Data, Virtual Event, China, June 20-25, 2021*, G. Li, Z. Li, S. Idreos and D. Srivastava, eds, ACM, 2021, pp. 2290–2299. doi:10.1145/3448016.3457323.

[50] S. Sagawa, P.W. Koh, T.B. Hashimoto and P. Liang, Distributionally Robust Neural Networks for Group Shifts: On the Importance of Regularization for Worst-Case Generalization, in: *8th International Conference on Learning Representations, ICLR 2020*, 2020. https://www.scopus.com/inward/record.uri?eid=2-s2.0-85150613153&partnerID=40&md5=9500e6ad6b1c2fc3da67c2f61683471f.

[51] M.K. Sarker, L. Zhou, A. Eberhart and P. Hitzler, Neuro-symbolic artificial intelligence, *AI Commun.* **34**(3) (2021), 197–209. doi:10.3233/AIC-210084.

[52] F. Shakerin, E. Salazar and G. Gupta, A New Algorithm to Automate Inductive Learning of Default Theories, *Theory Pract. Log. Program.* **17**(5–6) (2017), 1010–1026. doi:10.1017/S1471068417000333.

[53] H. Shindo, V. Pfanschilling, D.S. Dhami and K. Kersting, αILP: thinking visual scenes as differentiable logic programs, *Mach. Learn.* **112**(5) (2023), 1465–1497. doi:10.1007/S10994-023-06320-1. https://doi.org/10.1007/s10994-023-06320-1.

[54] E. Slyman, M. Kahng and S. Lee, VLSlice: Interactive Vision-and-Language Slice Discovery, in: *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*, IEEE, 2023, pp. 15245–15255. doi:10.1109/ICCV51070.2023.01403.

[55] N.S. Sohoni, J. Dunnmon, G. Angus, A. Gu and C. Ré, No Subclass Left Behind: Fine-Grained Robustness in Coarse-Grained Classification Problems, in: *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, H. Larochelle,

M. Ranzato, R. Hadsell, M. Balcan and H. Lin, eds, 2020. https://proceedings.neurips.cc/paper/2020/hash/e0688d13958a19e087e123148555e4b4-Abstract.html.

[56] R. Szeliski, *Computer Vision - Algorithms and Applications, Second Edition*, Texts in Computer Science, Springer, 2022. ISBN 978-3-030-34371-2. doi:10.1007/978-3-030-34372-9.

[57] J. Tello, M. de la Cruz, T. Ribeiro, J. Fiérrez, A. Morales, R. Tolosana, C.L. Alonso and A. Ortega, Symbolic AI (LFIT) for XAI to handle biases, in: *Proc. 1st Workshop on Fairness and Bias in AI co-located with 26th European Conference on Artificial Intelligence (ECAI 2023), Kraków, Poland, October 1st, 2023*, R. Calegari, A.A. Tubella, G. González-Castañé, V. Dignum and M. Milano, eds, CEUR Workshop Proceedings, Vol. 3523, CEUR-WS.org, 2023. https://ceur-ws.org/Vol-3523/paper10.pdf.

[58] M. Turcotte, S.H. Muggleton and M.J.E. Sternberg, Application of Inductive Logic Programming to Discover Rules Governing the Three-Dimensional Topology of Protein Structure, in: *Inductive Logic Programming, 8th International Workshop, ILP-98, Madison, Wisconsin, USA, July 22-24, 1998, Proceedings*, D. Page, ed., LNCS, Vol. 1446, Springer, 1998, pp. 53–64. doi:10.1007/BFB0027310. https://doi.org/10.1007/BFb0027310.

[59] H. Wang and G. Gupta, FOLD-R++: A Scalable Toolset for Automated Inductive Learning of Default Theories from Mixed Data, in: *Functional and Logic Programming - 16th International Symposium, FLOPS 2022, Kyoto, Japan, May 10-12, 2022, Proceedings*, M. Hanus and A. Igarashi, eds, LNCS, Vol. 13215, Springer, 2022, pp. 224–242. doi:10.1007/978-3-030-99461-7_13.

[60] J. Wu, W. Gan, Z. Chen, S. Wan and P.S. Yu, Multimodal Large Language Models: A Survey, in: *IEEE International Conference on Big Data, BigData 2023, Sorrento, Italy, December 15-18, 2023*, J. He, T. Palpanas, X. Hu, A. Cuzzocrea, D. Dou, D. Slezak, W. Wang, A. Gruca, J.C. Lin and R. Agrawal, eds, IEEE, 2023, pp. 2247–2256. doi:10.1109/BIGDATA59044.2023.10386743. https://doi.org/10.1109/BigData59044.2023.10386743.

[61] R. Yan, T. Ma, A. Fokoue, M. Chang and A. Julius, Neuro-symbolic Models for Interpretable Time Series Classification using Temporal Logic Description, in: *IEEE International Conference on Data Mining, ICDM 2022, Orlando, FL, USA, November 28 - Dec. 1, 2022*, X. Zhu, S. Ranka, M.T. Thai, T. Washio and X. Wu, eds, IEEE, 2022, pp. 618–627. doi:10.1109/ICDM54844.2022.00072.

[62] Y.M. Youssef and M.E. Müller, A Review of Inductive Logic Programming Applications for Robotic Systems, in: *Inductive Logic Programming - 32nd International Conference, ILP 2023, Bari, Italy, November 13-15, 2023, Proceedings*, E. Bellodi, F.A. Lisi and R. Zese, eds, LNCS, Vol. 14363, Springer, 2023, pp. 154–165. doi:10.1007/978-3-031-49299-0_11.

[63] M. Zhang, Y. Zhang, L. Zhang, C. Liu and S. Khurshid, DeepRoad: GAN-Based Metamorphic Testing and Input Validation Framework for Autonomous Driving Systems, in: *Proc. 33rd ACM/IEEE International Conference on Automated Software Engineering, ASE 2018, Montpellier, France, September 3-7, 2018*, M. Huchard, C. Kästner and G. Fraser, eds, ACM, 2018, pp. 132–142. doi:10.1145/3238147.3238187.

[64] W. Zhang, Y. Liu, Z. Wang and J. Wang, Learning to Binarize Continuous Features for Neuro-Rule Networks, in: *Proc. Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI 2023, 19th-25th August 2023, Macao, SAR, China*, ijcai.org, 2023, pp. 4584–4592. doi:10.24963/IJCAI.2023/510. https://doi.org/10.24963/ijcai.2023/510.

Appendix A. Taxonomies

A.1. Super-CLEVR – Vehicle Type

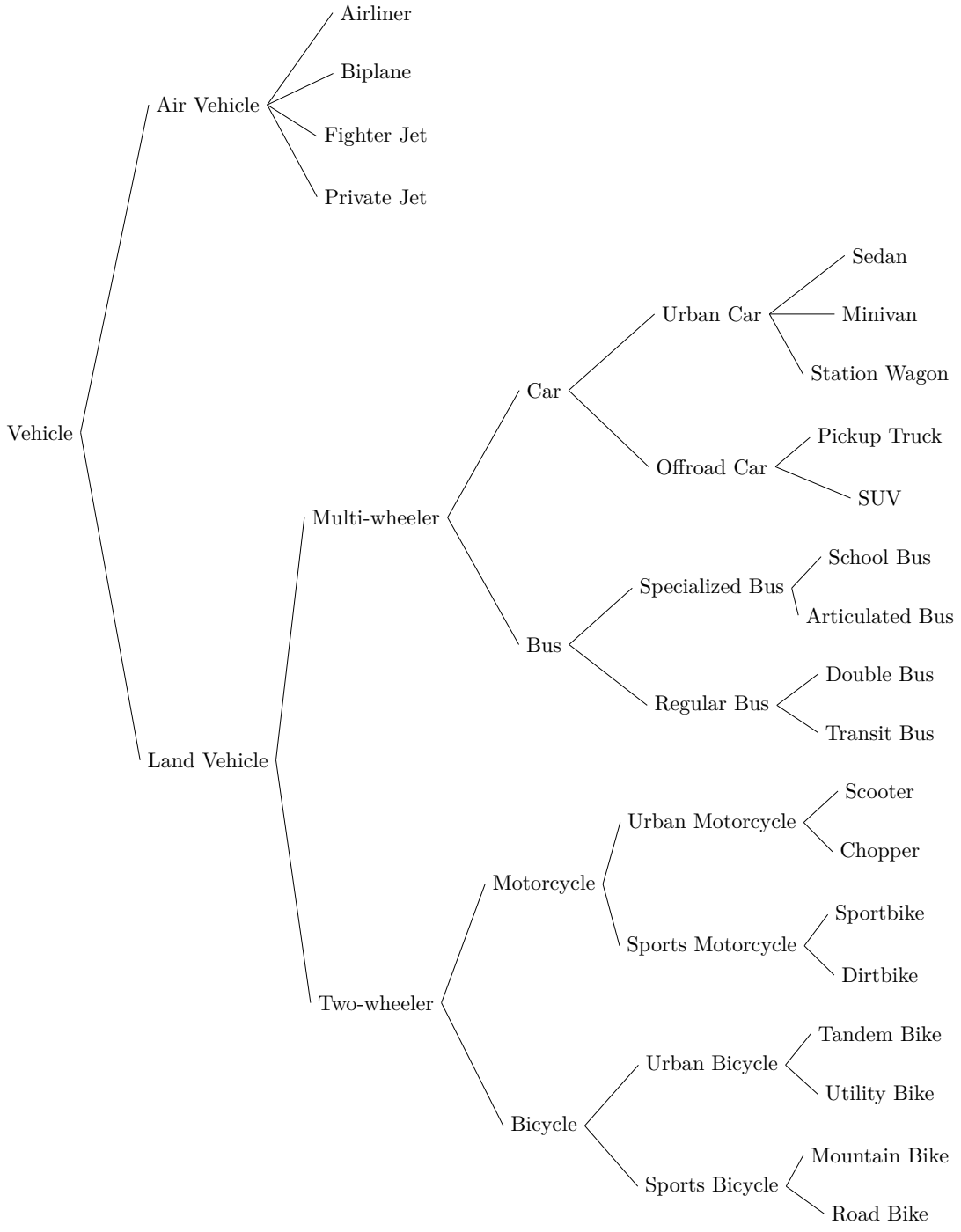

Fig. 7. *Super-CLEVR Vehicle Type* taxonomy.

A.2. Super-CLEVR – Primary Purpose

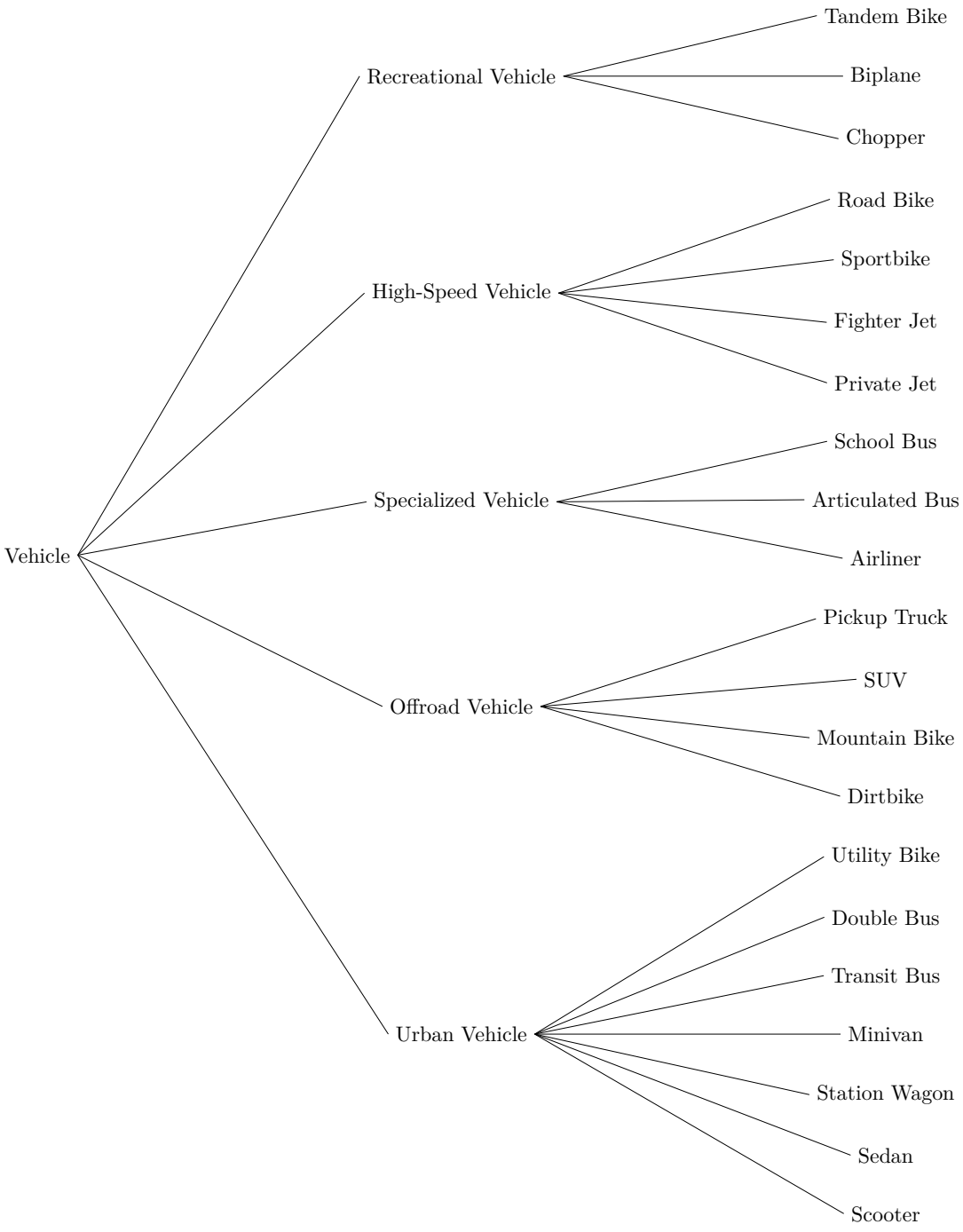

Fig. 8. *Super-CLEVR Primary Purpose* taxonomy.

A.3. ImageNet – Vehicle

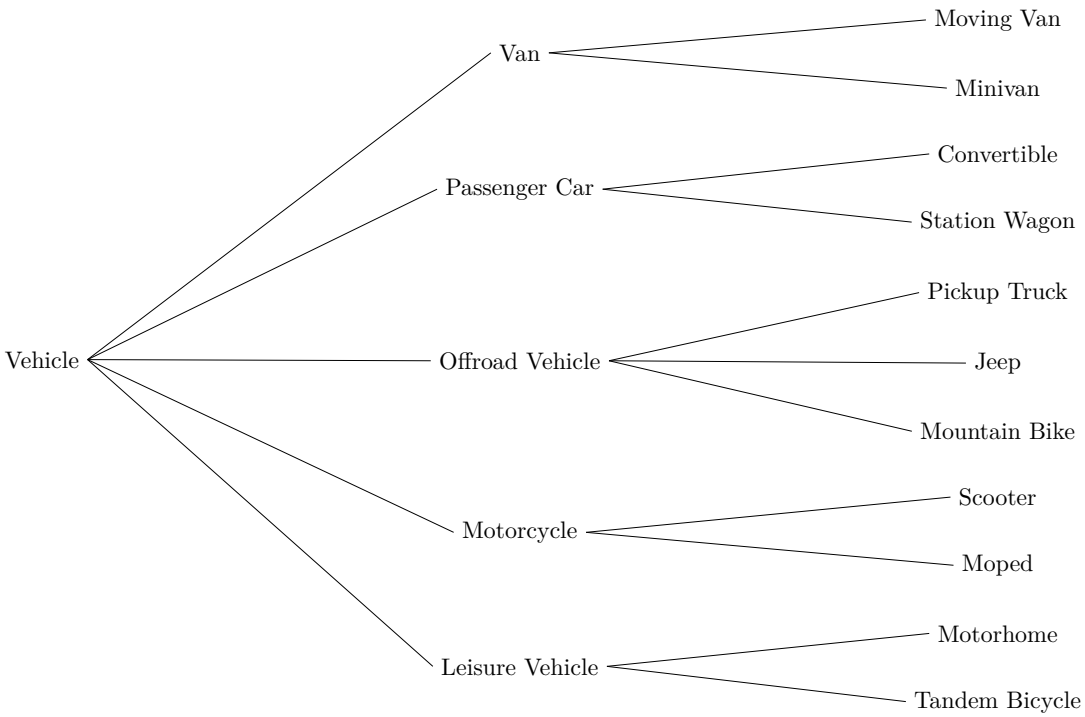

Fig. 9. *ImageNet Vehicle* taxonomy.

Appendix B. ILP Encodings

This section details the specific ILP encoding formats required by the different systems used in our experiments: *Popper*, *FOLD-R++*, and *FastLAS*. Each system has its own syntax to represent examples, background knowledge, and mode bias. We provide encoding extracts for two concrete scenarios:

- the "urban bicycle" class of the first iteration on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*;
- the "leisure vehicle" class of the first iteration on the *VE:$\mathcal{H}1$* model for *ImageNet*.

The complete encodings used in our experiments are available in the online repository.

B.1. Popper Encoding

Popper utilises the syntax of Prolog. It requires three main components: a set of positive and negative examples, background knowledge, and mode bias that defines the structure of learnable rules.

B.1.1. Super-CLEVR – VT:$\mathcal{H}4$

Popper encoding for the "urban bicycle" class at the first iteration on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*:

Examples. Positive (`pos`) and negative (`neg`) examples identify scenes that are misclassified (`hard`) or correctly classified, respectively.

```
neg(hard(s1)).
...
pos(hard(s19)).
...
```

Background Knowledge. A set of ground facts describing all scenes and the objects they contain. Each fact relates a scene or object ID to a specific property.

```
contains(s1, o0_1).
brown(o0_1).
southwest(o0_1).
rubber(o0_1).
pickup(o0_1).
small(o0_1).
...
contains(s19, o0_19).
yellow(o0_19).
south(o0_19).
metal(o0_19).
utility(o0_19).
large(o0_19).
...
```

Mode Bias. These declarations define the hypothesis space. `head_pred` specifies the target predicate for the rule head, while `body_pred` lists all admissible predicates for the rule body.

```
% Rule heads
head_pred(hard,1). type(hard,(scene_type,)). direction(hard,(in,)).

% Rule bodies
body_pred(contains,2). type(contains,(scene_type,obj_type)).
direction(contains,(in,out)).
```

```
%% shapes
body_pred(private,1). type(private,(obj_type,)). direction(private,(in,)).
body_pred(fighter,1). type(fighter,(obj_type,)). direction(fighter,(in,)).
body_pred(biplane,1). type(biplane,(obj_type,)). direction(biplane,(in,)).
body_pred(airliner,1). type(airliner,(obj_type,)). direction(airliner,(in,)).
body_pred(road,1). type(road,(obj_type,)). direction(road,(in,)).
body_pred(utility,1). type(utility,(obj_type,)). direction(utility,(in,)).
body_pred(tandem,1). type(tandem,(obj_type,)). direction(tandem,(in,)).
body_pred(mountain,1). type(mountain,(obj_type,)). direction(mountain,(in,)).
body_pred(articulated,1). type(articulated,(obj_type,)). direction(articulated,(in,)).
body_pred(transit,1). type(transit,(obj_type,)). direction(transit,(in,)).
body_pred(double,1). type(double,(obj_type,)). direction(double,(in,)).
body_pred(school,1). type(school,(obj_type,)). direction(school,(in,)).
body_pred(suv,1). type(suv,(obj_type,)). direction(suv,(in,)).
body_pred(wagon,1). type(wagon,(obj_type,)). direction(wagon,(in,)).
body_pred(minivan,1). type(minivan,(obj_type,)). direction(minivan,(in,)).
body_pred(sedan,1). type(sedan,(obj_type,)). direction(sedan,(in,)).
body_pred(pickup,1). type(pickup,(obj_type,)). direction(pickup,(in,)).
body_pred(chopper,1). type(chopper,(obj_type,)). direction(chopper,(in,)).
body_pred(dirtbike,1). type(dirtbike,(obj_type,)). direction(dirtbike,(in,)).
body_pred(scooter,1). type(scooter,(obj_type,)). direction(scooter,(in,)).
body_pred(sportbike,1). type(sportbike,(obj_type,)). direction(sportbike,(in,)).
%% colors
body_pred(gray,1). type(gray,(obj_type,)). direction(gray,(in,)).
body_pred(red,1). type(red,(obj_type,)). direction(red,(in,)).
body_pred(blue,1). type(blue,(obj_type,)). direction(blue,(in,)).
body_pred(green,1). type(green,(obj_type,)). direction(green,(in,)).
body_pred(brown,1). type(brown,(obj_type,)). direction(brown,(in,)).
body_pred(purple,1). type(purple,(obj_type,)). direction(purple,(in,)).
body_pred(cyan,1). type(cyan,(obj_type,)). direction(cyan,(in,)).
body_pred(yellow,1). type(yellow,(obj_type,)). direction(yellow,(in,)).
%% sizes
body_pred(small,1). type(small,(obj_type,)). direction(small,(in,)).
body_pred(large,1). type(large,(obj_type,)). direction(large,(in,)).
%% directions
body_pred(east,1). type(east,(obj_type,)). direction(east,(in,)).
body_pred(northeast,1). type(northeast,(obj_type,)). direction(northeast,(in,)).
body_pred(north,1). type(north,(obj_type,)). direction(north,(in,)).
body_pred(northwest,1). type(northwest,(obj_type,)). direction(northwest,(in,)).
body_pred(west,1). type(west,(obj_type,)). direction(west,(in,)).
body_pred(southwest,1). type(southwest,(obj_type,)). direction(southwest,(in,)).
body_pred(south,1). type(south,(obj_type,)). direction(south,(in,)).
body_pred(southeast,1). type(southeast,(obj_type,)). direction(southeast,(in,)).
%% materials
body_pred(rubber,1). type(rubber,(obj_type,)). direction(rubber,(in,)).
body_pred(metal,1). type(metal,(obj_type,)). direction(metal,(in,)).
```

B.1.2. ImageNet – VE:$\mathcal{H}$1

Popper encoding for the "leisure vehicle" class at the first iteration on the *VE:$\mathcal{H}$1* model for *ImageNet*:

Examples.

```
pos(hard(s0)).
...
neg(hard(s6)).
...
```

Background Knowledge.

```
contains(s0, o0_0).
black(o0_0).
side_view(o0_0).
foreground(o0_0).
tandem_bicycle(o0_0).
fully_visible(o0_0).
rural_outdoor(s0).
false(s0).
daytime(s0).
person(s0, 2).
...
contains(s6, o0_6).
blue(o0_6).
side_view(o0_6).
foreground(o0_6).
tandem_bicycle(o0_6).
fully_visible(o0_6).
rural_outdoor(s6).
false(s6).
daytime(s6).
person(s6, 0).
...
```

Mode Bias.

```
% Rule heads
head_pred(hard,1). type(hard,(scene_type,)). direction(hard,(in,)).

% Rule bodies
body_pred(contains,2). type(contains,(scene_type,obj_type)).
direction(contains,(in,out)).

%% person
body_pred(person,2). type(person,(scene_type,int)). direction(person,(in,out)).
%% background
body_pred(indoor,1). type(indoor,(scene_type,)). direction(indoor,(in,)).
body_pred(urban_outdoor,1). type(urban_outdoor,(scene_type,)).
direction(urban_outdoor,(in,)).
body_pred(rural_outdoor,1). type(rural_outdoor,(scene_type,)).
direction(rural_outdoor,(in,)).
body_pred(plain,1). type(plain,(scene_type,)). direction(plain,(in,)).
%% snow
body_pred(false,1). type(false,(scene_type,)). direction(false,(in,)).
body_pred(true,1). type(true,(scene_type,)). direction(true,(in,)).
%% time_of_day
```

```
body_pred(daytime,1). type(daytime,(scene_type,)). direction(daytime,(in,)).
body_pred(nighttime,1). type(nighttime,(scene_type,)). direction(nighttime,(in,)).
%% color
body_pred(black,1). type(black,(obj_type,)). direction(black,(in,)).
body_pred(white,1). type(white,(obj_type,)). direction(white,(in,)).
body_pred(gray,1). type(gray,(obj_type,)). direction(gray,(in,)).
body_pred(red,1). type(red,(obj_type,)). direction(red,(in,)).
body_pred(blue,1). type(blue,(obj_type,)). direction(blue,(in,)).
body_pred(yellow,1). type(yellow,(obj_type,)). direction(yellow,(in,)).
body_pred(green,1). type(green,(obj_type,)). direction(green,(in,)).
body_pred(purple,1). type(purple,(obj_type,)). direction(purple,(in,)).
body_pred(orange,1). type(orange,(obj_type,)). direction(orange,(in,)).
body_pred(brown,1). type(brown,(obj_type,)). direction(brown,(in,)).
body_pred(multicolor,1). type(multicolor,(obj_type,)). direction(multicolor,(in,)).
%% orientation
body_pred(rear_view,1). type(rear_view,(obj_type,)). direction(rear_view,(in,)).
body_pred(front_view,1). type(front_view,(obj_type,)). direction(front_view,(in,)).
body_pred(side_view,1). type(side_view,(obj_type,)). direction(side_view,(in,)).
body_pred(top_view,1). type(top_view,(obj_type,)). direction(top_view,(in,)).
%% position
body_pred(foreground,1). type(foreground,(obj_type,)). direction(foreground,(in,)).
body_pred(background,1). type(background,(obj_type,)). direction(background,(in,)).
%% type
body_pred(minivan,1). type(minivan,(obj_type,)). direction(minivan,(in,)).
body_pred(moving_van,1). type(moving_van,(obj_type,)). direction(moving_van,(in,)).
body_pred(station_wagon,1). type(station_wagon,(obj_type,)).
direction(station_wagon,(in,)).
body_pred(convertible,1). type(convertible,(obj_type,)). direction(convertible,(in,)).
body_pred(mountain_bike,1). type(mountain_bike,(obj_type,)).
direction(mountain_bike,(in,)).
body_pred(jeep,1). type(jeep,(obj_type,)). direction(jeep,(in,)).
body_pred(pickup_truck,1). type(pickup_truck,(obj_type,)).
direction(pickup_truck,(in,)).
body_pred(tandem_bicycle,1). type(tandem_bicycle,(obj_type,)).
direction(tandem_bicycle,(in,)).
body_pred(motorhome,1). type(motorhome,(obj_type,)). direction(motorhome,(in,)).
body_pred(moped,1). type(moped,(obj_type,)). direction(moped,(in,)).
body_pred(scooter,1). type(scooter,(obj_type,)). direction(scooter,(in,)).
%% visibility
body_pred(fully_visible,1). type(fully_visible,(obj_type,)).
direction(fully_visible,(in,)).
body_pred(partially_visible,1). type(partially_visible,(obj_type,)).
direction(partially_visible,(in,)).
```

B.2. FOLD-R++ Encoding

FOLD-R++ takes tabular data as input, specified in the text data format *Comma-Separated Values (CSV)*. Each row corresponds to a single scene (example), and each column represents a specific feature. Object properties within a scene are flattened into distinct columns (e.g., `shape_obj0`, `color_obj0`, `size_obj0`, etc.).

B.2.1. Super-CLEVR – VT:$\mathcal{H}4$

FOLD-R++ encoding for the "urban bicycle" class at the first iteration on the *VT:*$\mathcal{H}4$ model for *Super-CLEVR*. The feature `label` indicates `hard` (misclassified) or `easy` (correctly classified) examples, and subsequent columns describe the properties of each object (`obj0`, `obj1`, ...) in the scene.

```
id,label,shape_obj0,color_obj0,size_obj0,direction_obj0,material_obj0,shape_obj1,...
1,easy,pickup,brown,small,southwest,rubber,suv,...
...
3601,hard,utility,yellow,large,south,metal,fighter,...
...
```

B.2.2. ImageNet – VE:$\mathcal{H}1$

FOLD-R++ encoding for the "leisure vehicle" class at the first iteration on the *VE:*$\mathcal{H}1$ model for *ImageNet*. Environment features like `person` count and `background` type appear alongside object features like `color_obj0` and `type_obj0`.

```
id,label,person,background,...,color_obj0,orientation_obj0,position_obj0,type_obj0,...
1,hard,2,rural_outdoor,...,black,side_view,foreground,tandem_bicycle,...
...
7,easy,0,rural_outdoor,...,blue,side_view,foreground,tandem_bicycle,...
...
```

B.3. FastLAS Encoding

FastLAS utilises the syntax of ASP. A key difference from *Popper* is that background knowledge is *context-dependent*, i.e. it is encapsulated within each example definition rather than provided globally. Positive (`#pos`) and negative (`#neg`) examples represent misclassified (`hard`) or correctly classified scenes, respectively.

B.3.1. Super-CLEVR – VT:$\mathcal{H}4$

FastLAS encoding for the "urban bicycle" class at the first iteration on the *VT:*$\mathcal{H}4$ model for *Super-CLEVR*:

Context-Dependent Examples. Each `#pos` or `#neg` block defines one example. The first argument is the unique example identifier plus its penalty, the second is the atom to be proved for the positive examples (or not proved for the negative ones), the third is the set of atoms that must not be proved (empty in our examples), and the fourth block contains all background facts relevant only to that example.

```
#neg(s1@2, {hard(1)}, {}, {
    contains(1, 0).
    color(0, brown).
    direction(0, southwest).
    material(0, rubber).
    shape(0, pickup).
    size(0, small).
    ...
}).

...

#pos(s19@4, {hard(19)}, {}, {
    contains(19, 0).
    color(0, yellow).
```

```
    direction(0, south).
    material(0, metal).
    shape(0, utility).
    size(0, large).
    ...
}).

...
```

Mode Bias. These declarations define the hypothesis space. `#modeh` specifies the target predicate for the rule head, while `#modeb` lists all admissible predicates for the rule body. `#maxv(N)` specifies the maximum number of variables in any rule. Type domains define the possible constant values. Finally, `#bias("penalty...")` statements define the scoring function that guides the search, e.g. by penalising complex rules or rewarding the use of certain predicates.

```
% Configuration options
#maxv(2).

% Rule heads
#modeh(hard(var(sce_id))).

% Rule bodies
#modeb(contains(var(sce_id), var(obj_id))).
#modeb(not contains(var(sce_id), var(obj_id))).
#modeb(shape(var(obj_id), const(shape))).
#modeb(color(var(obj_id), const(color))).
#modeb(size(var(obj_id), const(size))).
#modeb(not size(var(obj_id), const(size))).
#modeb(direction(var(obj_id), const(direction))).
#modeb(not direction(var(obj_id), const(direction))).
#modeb(material(var(obj_id), const(material))).
#modeb(not material(var(obj_id), const(material))).

% Type domains
sce_id(0..2499). obj_id(0..5).
%% shapes
shape(private). shape(fighter). shape(biplane). shape(airliner). shape(road).
shape(utility). shape(tandem). shape(mountain). shape(articulated). shape(transit).
shape(double). shape(school). shape(suv). shape(wagon). shape(minivan). shape(sedan).
shape(pickup). shape(chopper). shape(dirtbike). shape(scooter). shape(sportbike).
%% colors
color(gray). color(red). color(blue). color(green). color(brown). color(purple).
color(cyan). color(yellow).
%% sizes
size(small). size(large).
%% directions
direction(east). direction(northeast). direction(north). direction(northwest).
direction(west). direction(southwest). direction(south). direction(southeast).
%% materials
material(rubber). material(metal).
```

```
% Scoring function
#bias("penalty(20, head(X)) :- in_head(X).").
#bias("penalty(-1, body(X)) :- in_body(X), X = contains(_,_).").
```

B.3.2. ImageNet – VE:$\mathcal{H}1$

FastLAS encoding for the "leisure vehicle" class at the first iteration on the *VE:$\mathcal{H}1$* model for *ImageNet*:

Context-Dependent Examples.

```
#pos(s0@4, {hard(0)}, {}, {
    person(0, 2).
    contains(0, 0).
    color(0, black).
    orientation(0, side_view).
    position(0, foreground).
    type(0, tandem_bicycle).
    visibility(0, fully_visible).
    background(0, rural_outdoor).
    snow(0, false).
    time_of_day(0, daytime).
}).

...

#neg(s6@2, {hard(6)}, {}, {
    contains(6, 0).
    color(0, blue).
    orientation(0, side_view).
    position(0, foreground).
    type(0, tandem_bicycle).
    visibility(0, fully_visible).
    background(6, rural_outdoor).
    snow(6, false).
    time_of_day(6, daytime).
    person(6, 0).
}).

...
```

Mode Bias.

```
% Configuration options
#maxv(2).

% Rule heads
#modeh(hard(var(sce_id))).

% Rule bodies
#modeb(contains(var(sce_id), var(obj_id))).
#modeb(not contains(var(sce_id), var(obj_id))).
#modeb(person(var(sce_id), const(person))).
#modeb(background(var(sce_id), const(background))).
#modeb(not background(var(sce_id), const(background))).
```

```
#modeb(snow(var(sce_id), const(snow))).
#modeb(time_of_day(var(sce_id), const(time_of_day))).
#modeb(color(var(obj_id), const(color))).
#modeb(orientation(var(obj_id), const(orientation))).
#modeb(not orientation(var(obj_id), const(orientation))).
#modeb(position(var(obj_id), const(position))).
#modeb(type(var(obj_id), const(type))).
#modeb(visibility(var(obj_id), const(visibility))).

% Type domains
sce_id(0..399). obj_id(0..4).
%% person
person(0..10).
%% background
background(indoor). background(urban_outdoor). background(rural_outdoor).
background(plain).
%% snow
snow(false). snow(true).
%% time_of_day
time_of_day(daytime). time_of_day(nighttime).
%% color
color(black). color(white). color(gray). color(red). color(blue). color(yellow).
color(green). color(purple). color(orange). color(brown). color(multicolor).
%% orientation
orientation(rear_view). orientation(front_view). orientation(side_view).
orientation(top_view).
%% position
position(foreground). position(background).
%% type
type(minivan). type(moving_van). type(station_wagon). type(convertible).
type(mountain_bike). type(jeep). type(pickup_truck). type(tandem_bicycle).
type(motorhome). type(moped). type(scooter).
%% visibility
visibility(fully_visible). visibility(partially_visible).

% Scoring function
#bias("penalty(10, head(X)) :- in_head(X).").
#bias("penalty(-1, body(X)) :- in_body(X), X = contains(_,_).").
#bias("penalty(-1, body(X)) :- in_body(X), X = type(_,_).").
```

Appendix C. Super-CLEVR Experiments

C.1. Vehicle Type: Hierarchy 1

C.1.1. Confusion Matrix

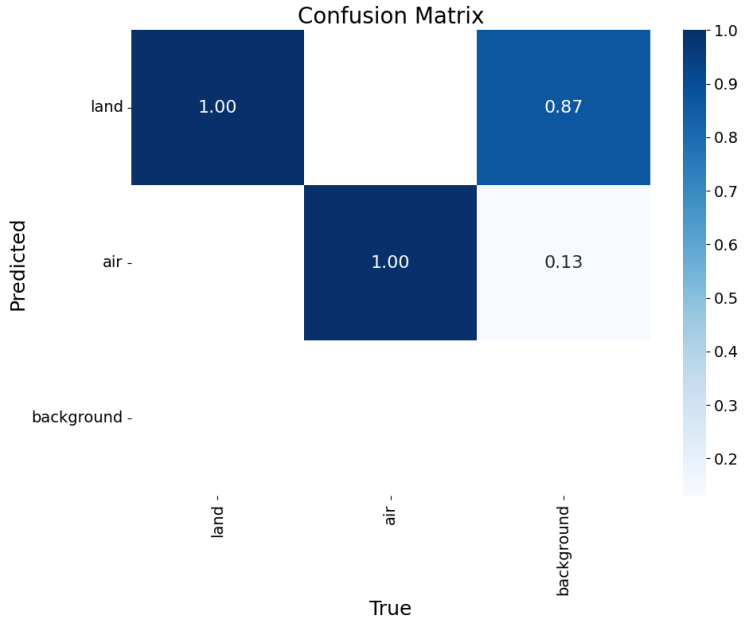

Fig. 10. Confusion matrix of the *VT:H1* model on the *Super-CLEVR* validation set after the initial model training.

C.1.2. F1-Confidence Curve

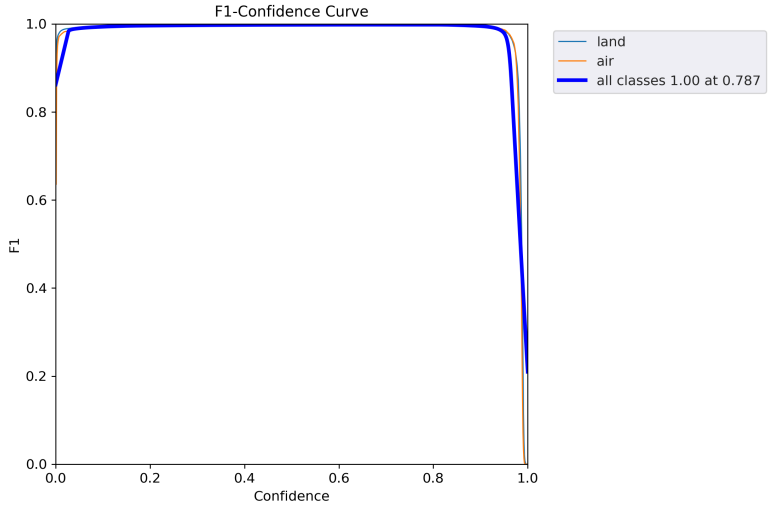

Fig. 11. F1-confidence curve of the *VT:H1* model on the *Super-CLEVR* validation set after the initial model training.

C.1.3. Model Training and Validation Performance Metrics

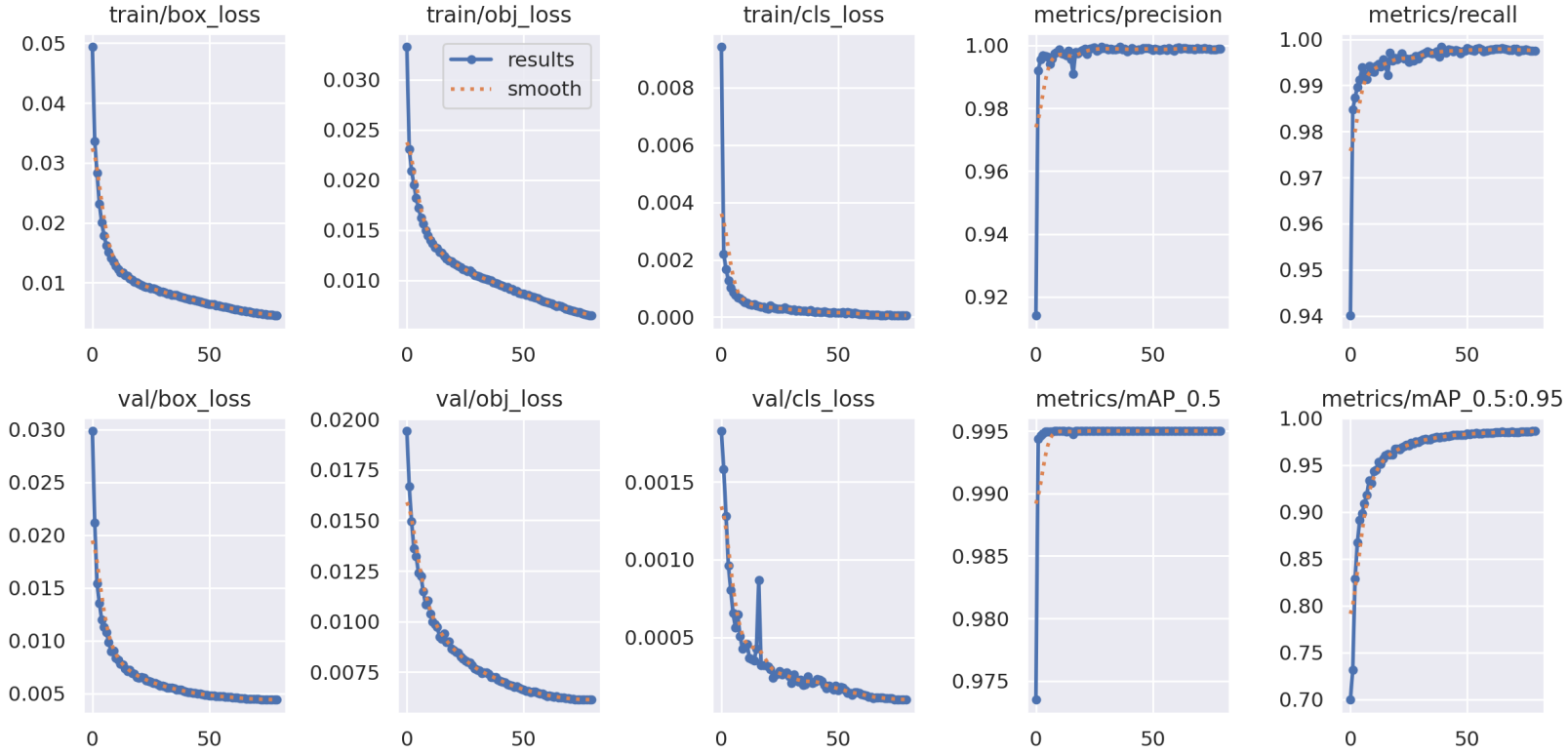

Fig. 12. *VT:$\mathcal{H}$1* model training and validation performance metrics on *Super-CLEVR* after the initial model training.

C.2. Vehicle Type: Hierarchy 2

C.2.1. Confusion Matrix

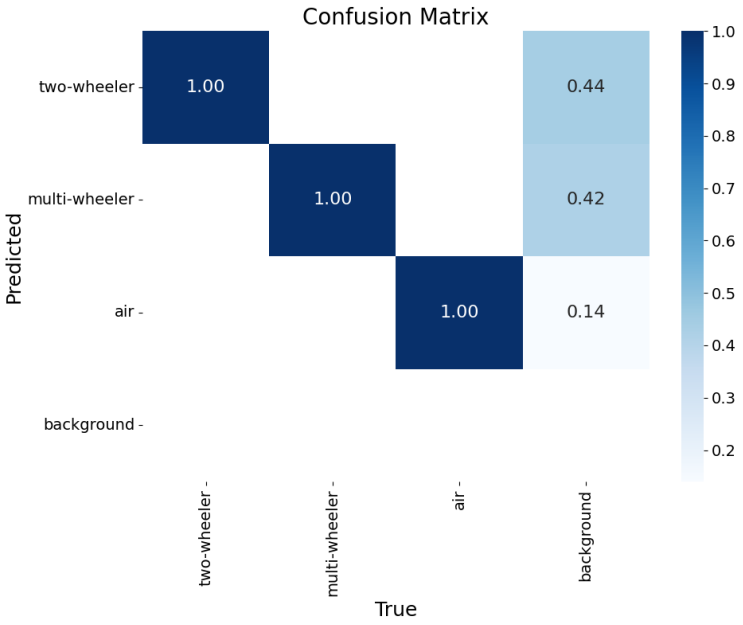

Fig. 13. Confusion matrix of the *VT:H2* model on the *Super-CLEVR* validation set after the initial model training.

C.2.2. F1-Confidence Curve

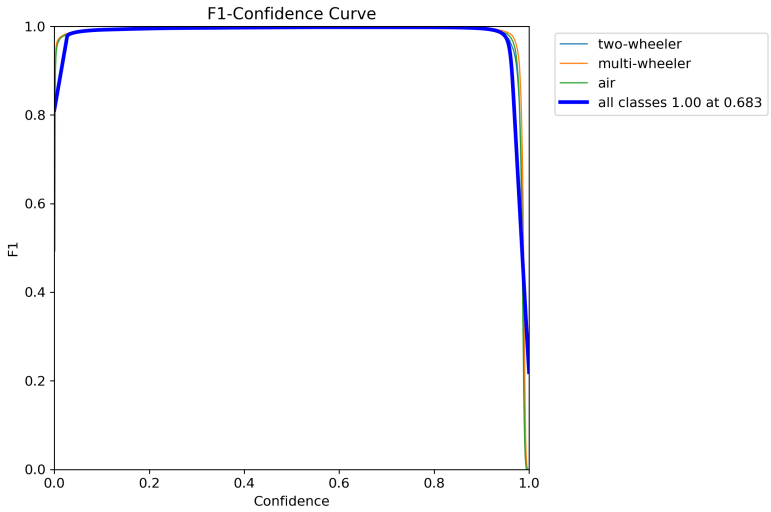

Fig. 14. F1-confidence curve of the *VT:H2* model on the *Super-CLEVR* validation set after the initial model training.

C.2.3. Model Training and Validation Performance Metrics

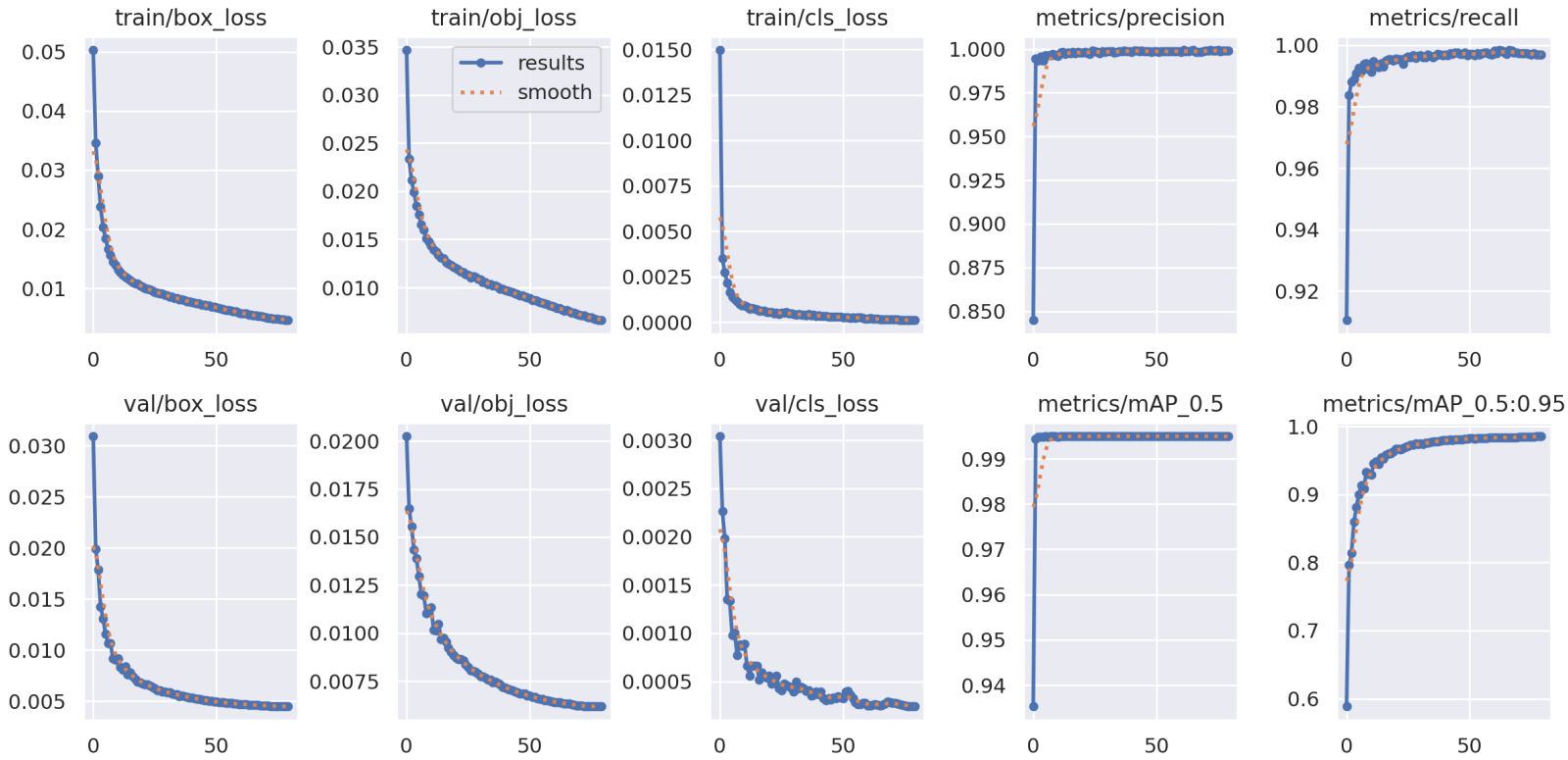

Fig. 15. *VT:$\mathcal{H}$2* model training and validation performance metrics on *Super-CLEVR* after the initial model training.

C.3. Vehicle Type: Hierarchy 3

C.3.1. Confusion Matrices

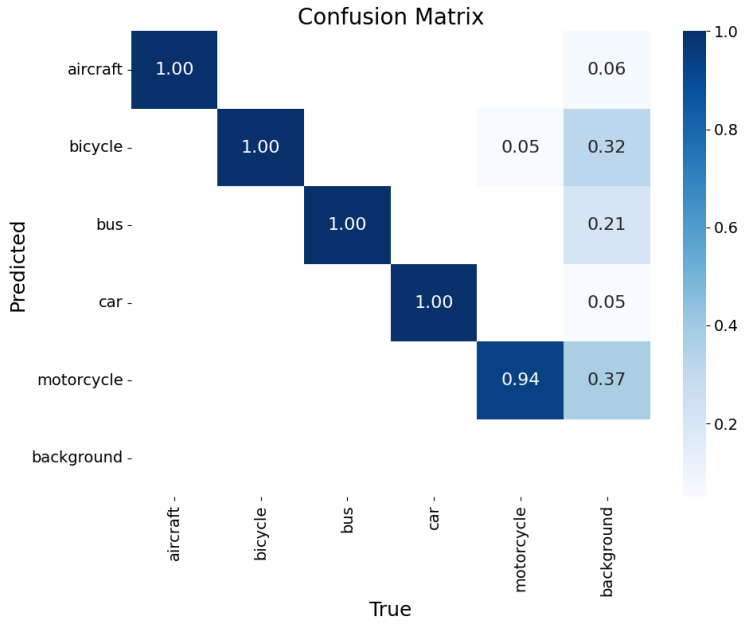

Fig. 16. Confusion matrix of the *VT:H3* model on the *Super-CLEVR* validation set after the initial model training.

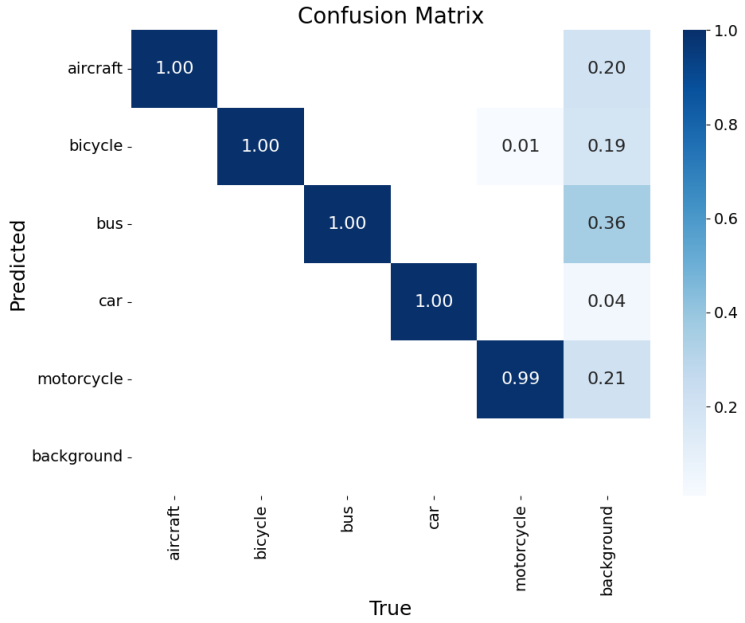

Fig. 17. Confusion matrix of the *VT:H3* model on the *Super-CLEVR* validation set after the model mending.

C.3.2. F1-Confidence Curves

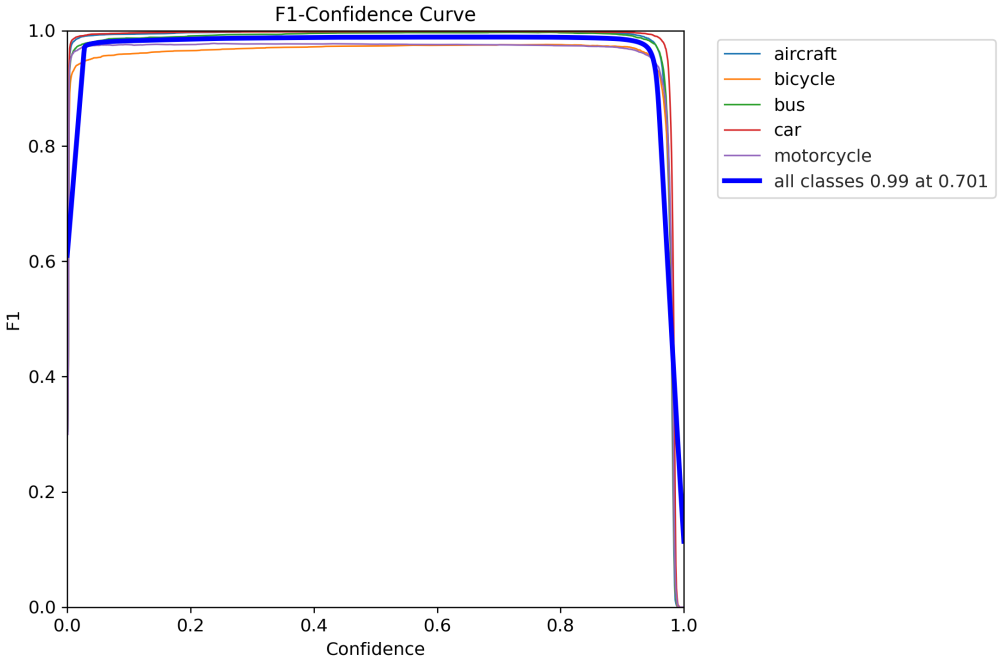

Fig. 18. F1-confidence curve of the *VT:H3* model on the *Super-CLEVR* validation set after the initial model training.

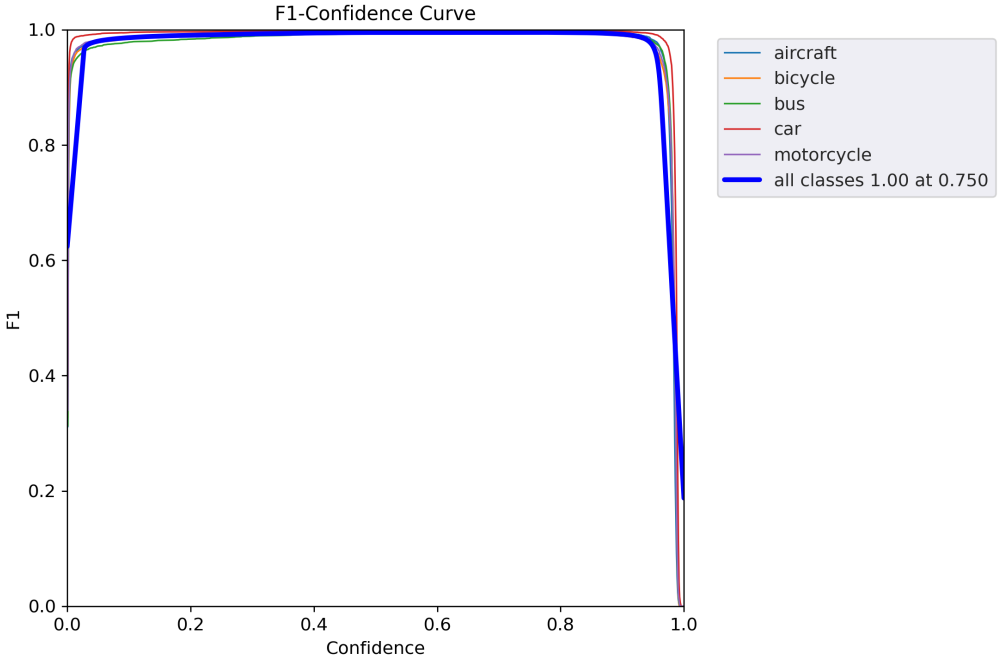

Fig. 19. F1-confidence curve of the *VT:H3* model on the *Super-CLEVR* validation set after the model mending.

C.3.3. Model Training and Validation Performance Metrics

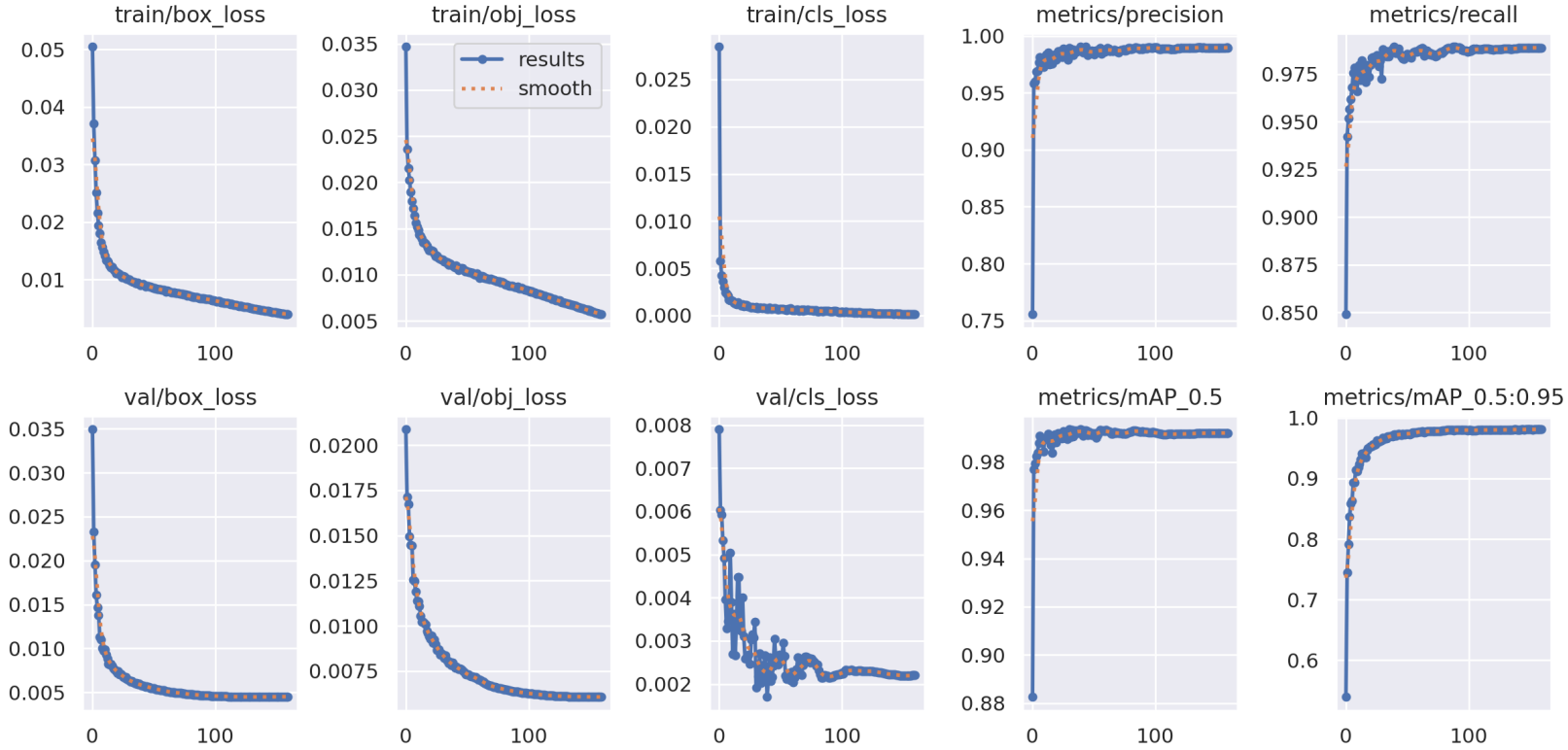

Fig. 20. *VT:ℋ3* model training and validation performance metrics on *Super-CLEVR* after the initial model training.

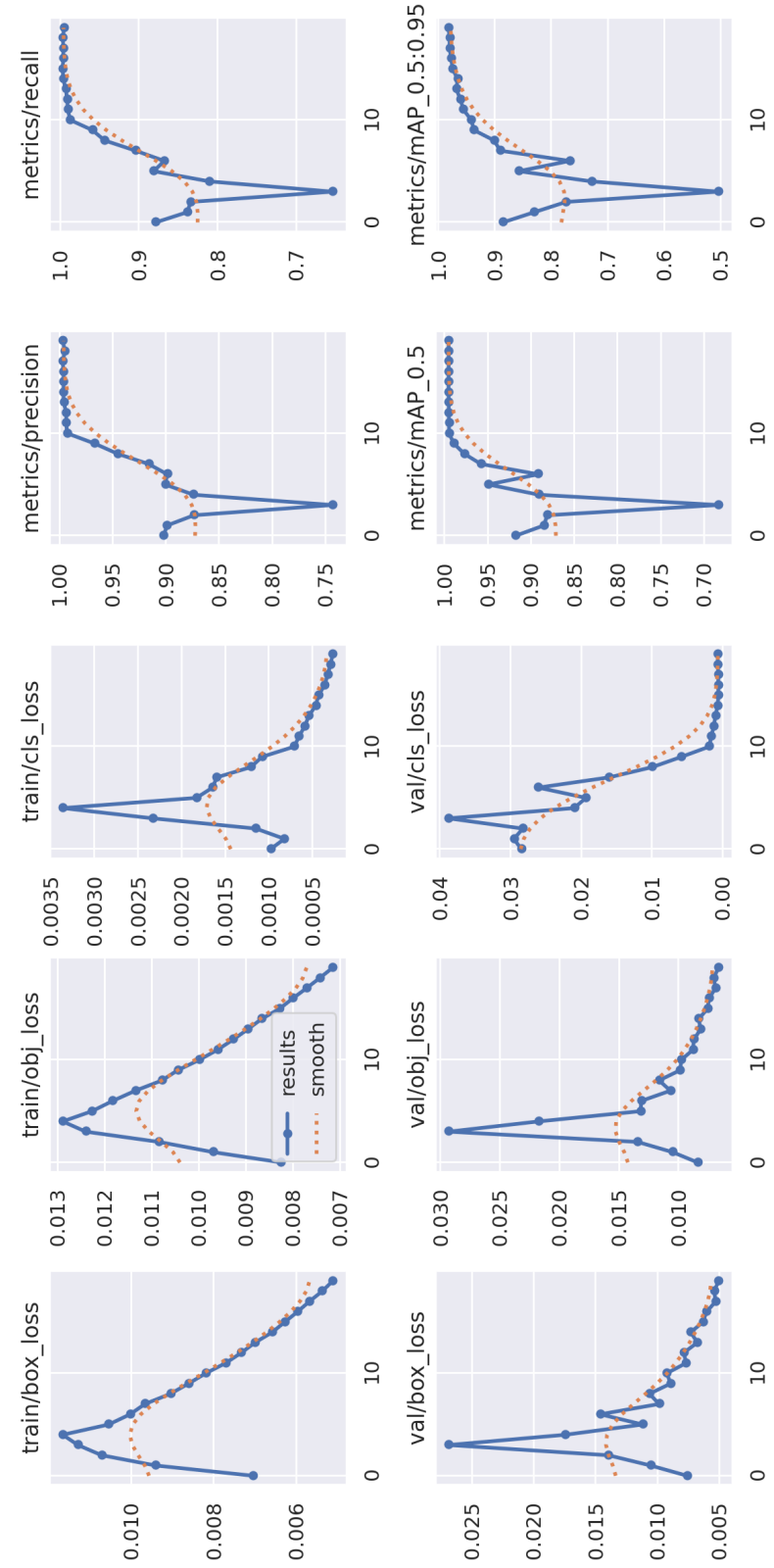

Fig. 21. *VT:H3* model training and validation performance metrics on *Super-CLEVR* after the model mending.

C.4. Vehicle Type: Hierarchy 4

C.4.1. Confusion Matrices

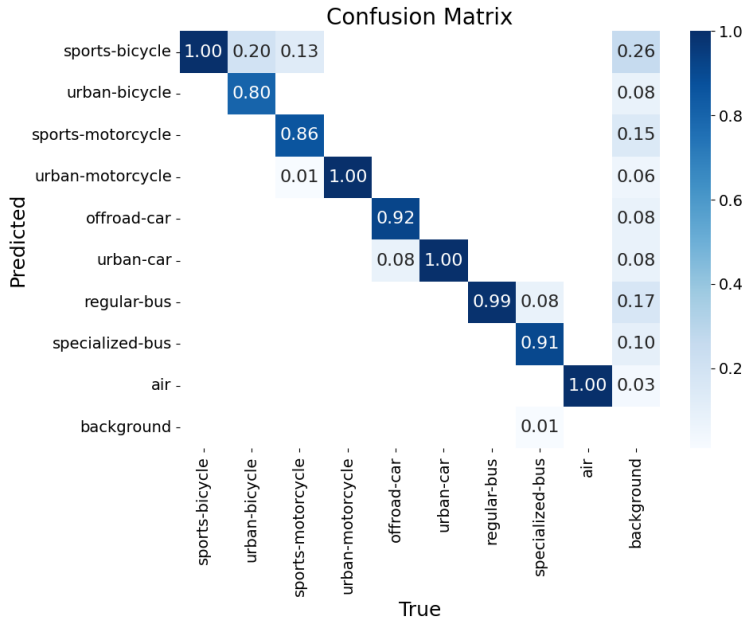

Fig. 22. Confusion matrix of the *VT:H4* model on the *Super-CLEVR* validation set after the initial model training.

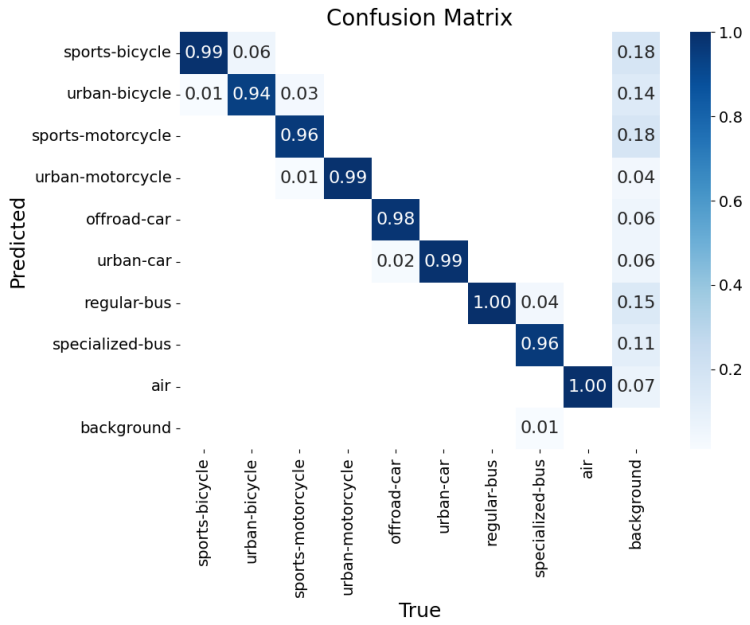

Fig. 23. Confusion matrix of the *VT:H4* model on the *Super-CLEVR* validation set after the first model mending iteration.

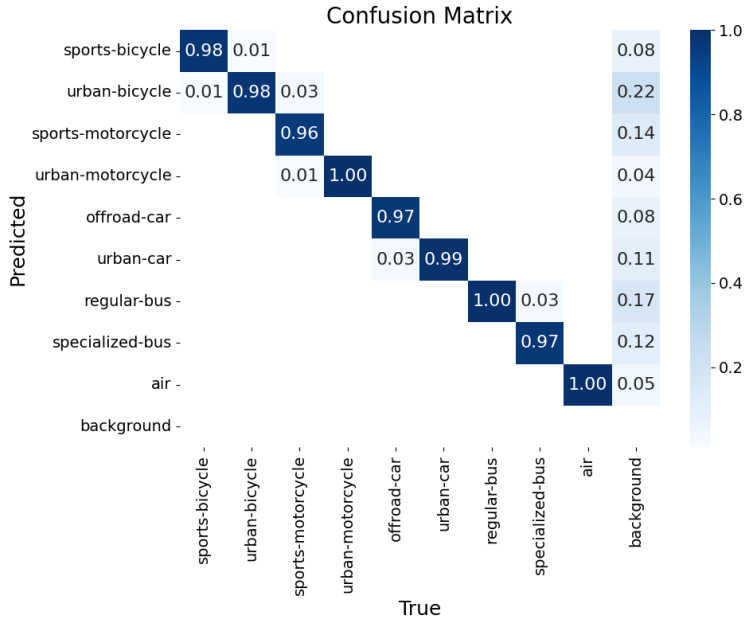

Fig. 24. Confusion matrix of the *VT:$\mathcal{H}$4* model on the *Super-CLEVR* validation set after the second model mending iteration.

C.4.2. F1-Confidence Curves

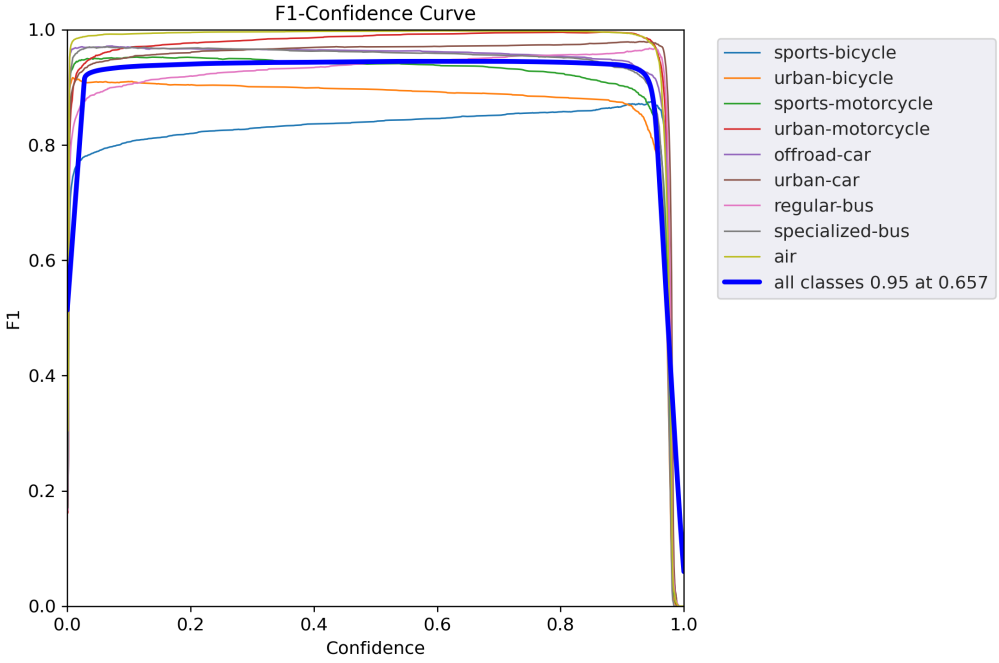

Fig. 25. F1-confidence curve of the *VT:$\mathcal{H}$4* model on the *Super-CLEVR* validation set after the initial model training.

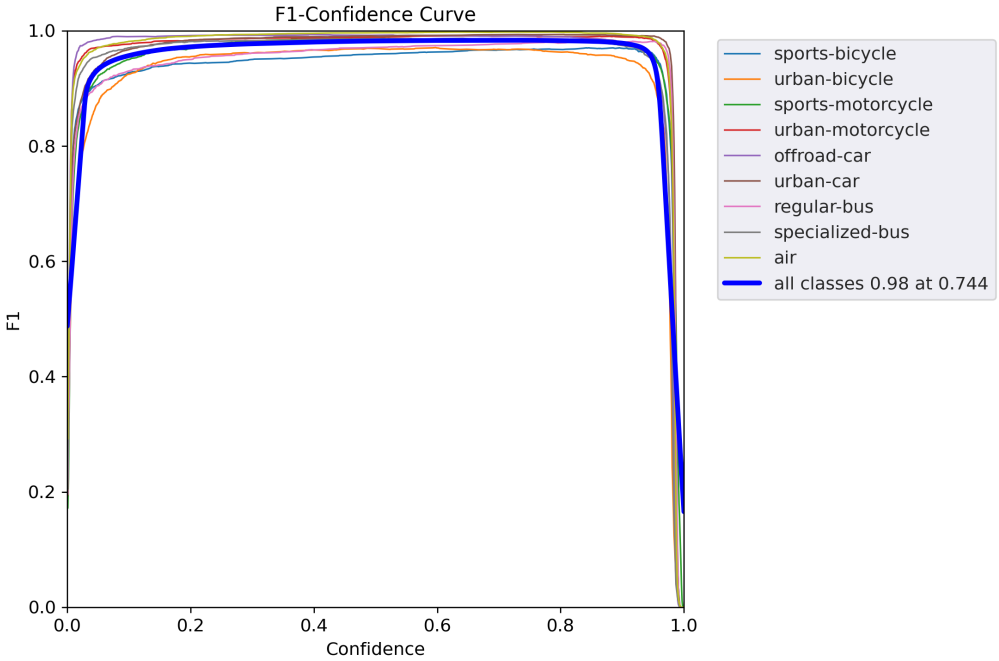

Fig. 26. F1-confidence curve of the *VT:$\mathcal{H}4$* model on the *Super-CLEVR* validation set after the first model mending iteration.

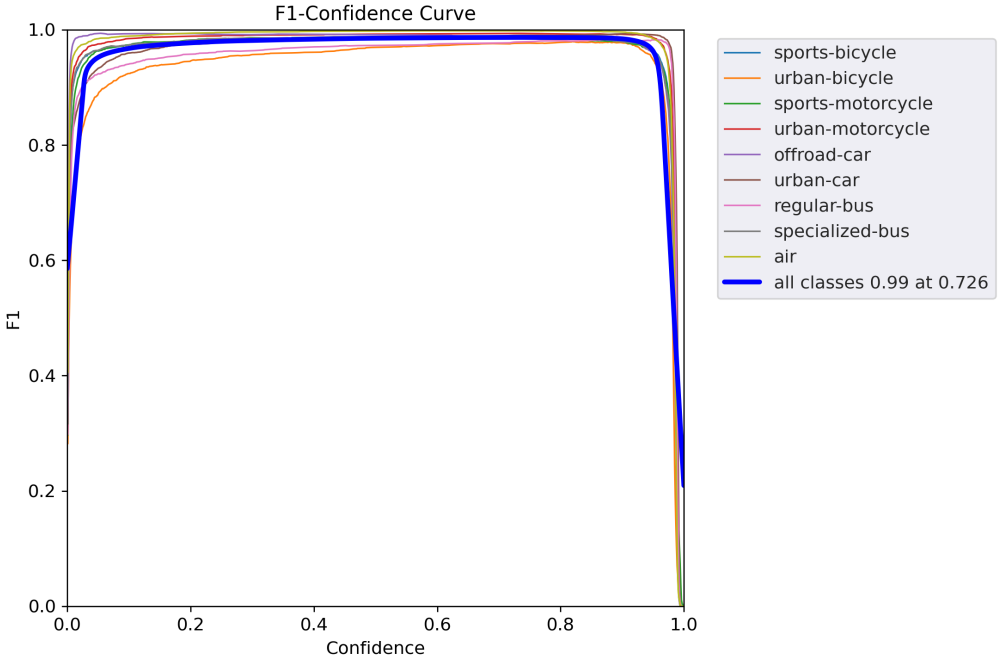

Fig. 27. F1-confidence curve of the *VT:$\mathcal{H}4$* model on the *Super-CLEVR* validation set after the second model mending iteration.

C.4.3. Model Training and Validation Performance Metrics

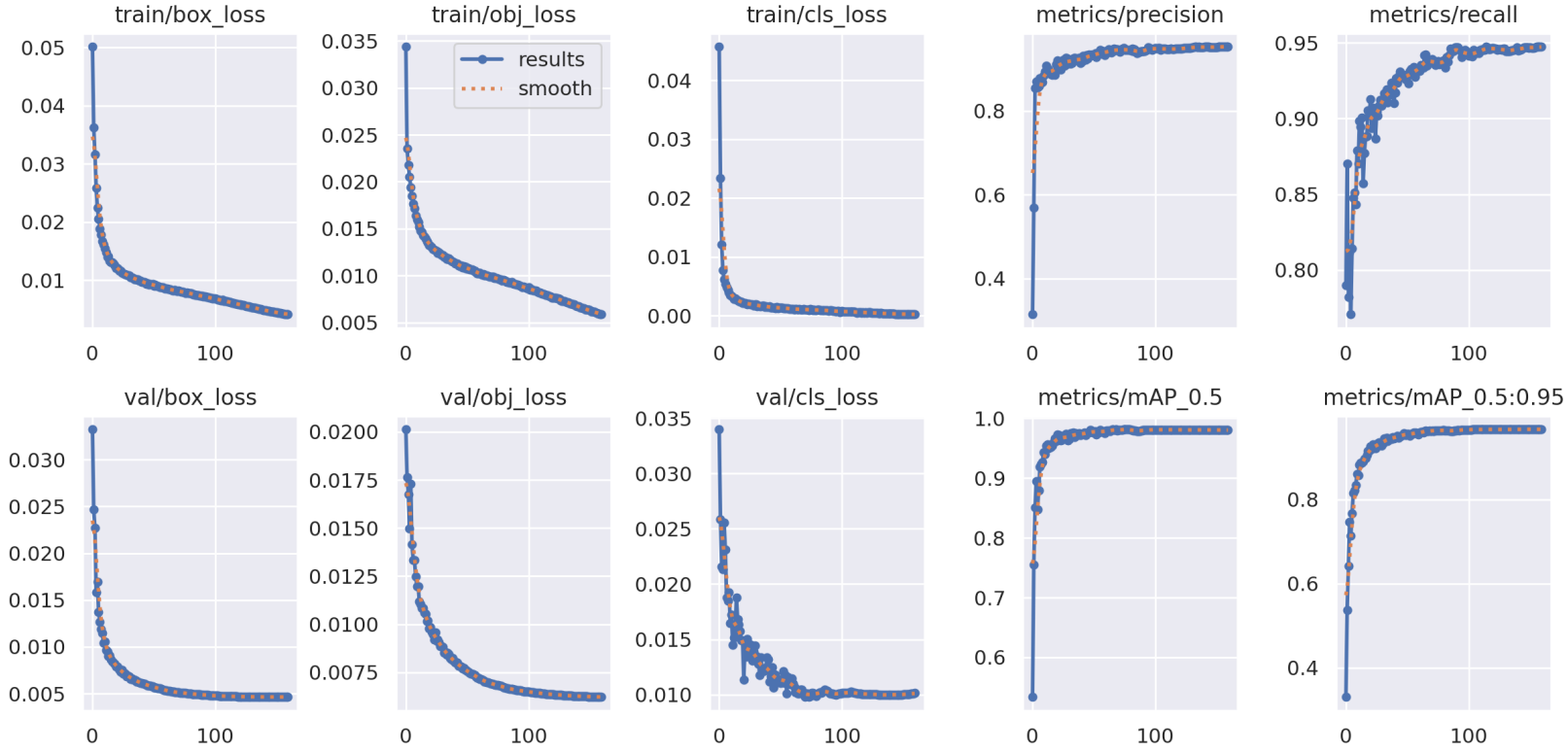

Fig. 28. *VT:$\mathcal{H}$4* model training and validation performance metrics on *Super-CLEVR* after the initial model training.

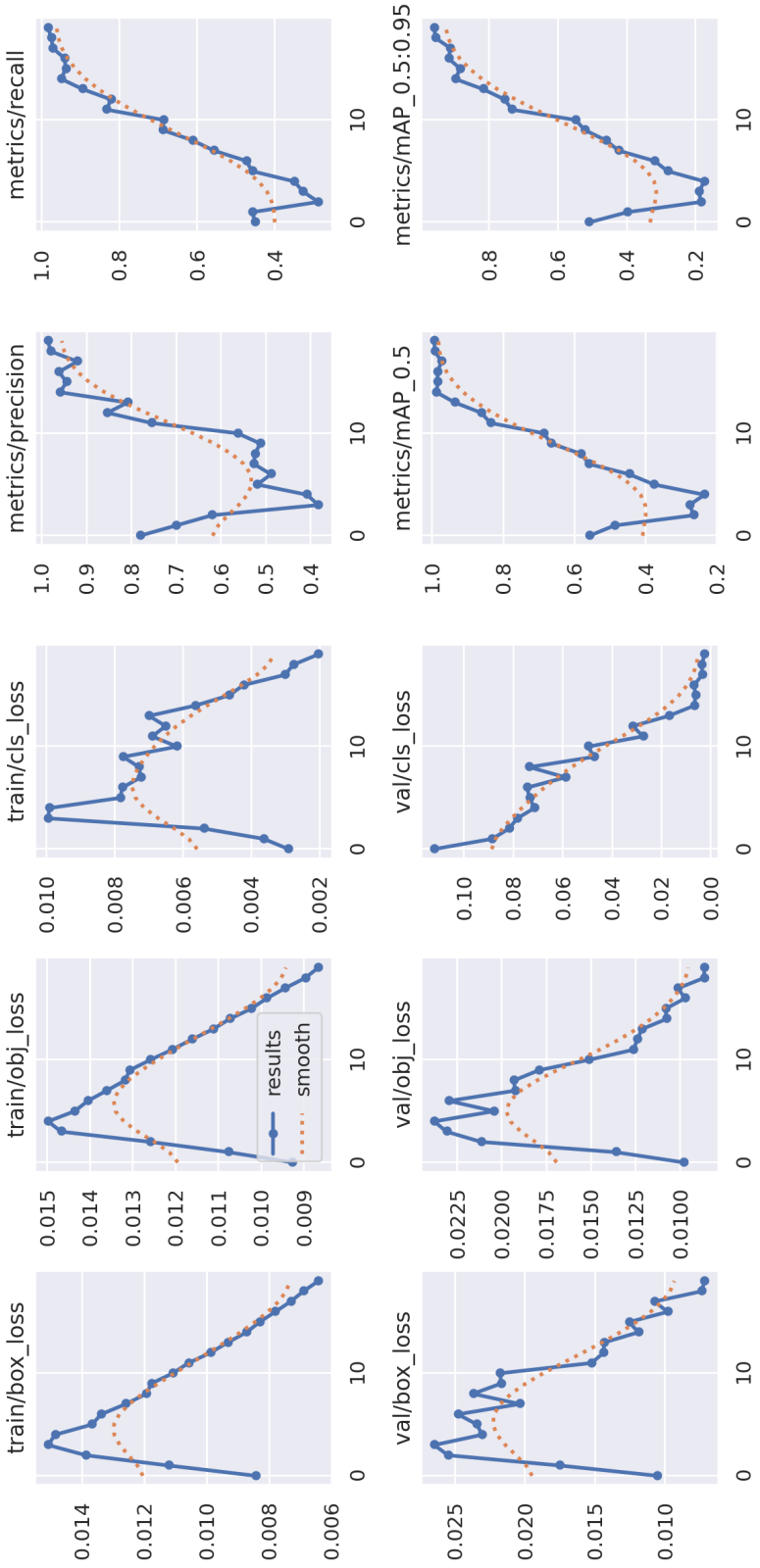

Fig. 29. *VT:H4* model training and validation performance metrics on *Super-CLEVR* after the first model mending iteration.

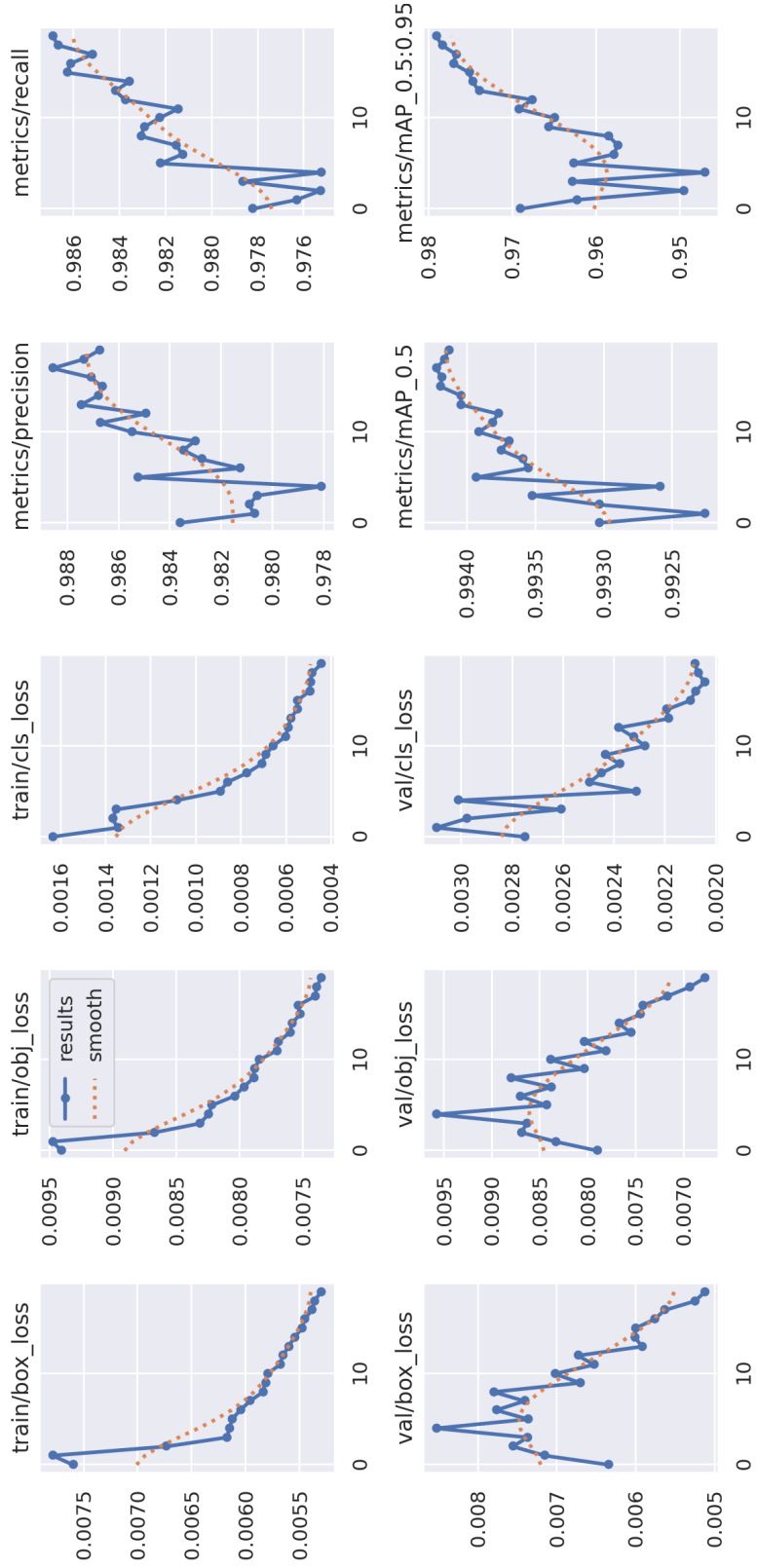

Fig. 30. *VT:H4* model training and validation performance metrics on *Super-CLEVR* after the second model mending iteration.

C.4.4. Rule Extraction and Selection Results

Table 18
Rule extraction results of the "offroad car" class of the first iteration on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	27.96	0	✗
FOLD-R++	2,686.69	24	**Pickup Truck**: **24** (**100.00**%) [**Rubber**: **24** (**100.00**%)], Mountain Bike: 9 (37.50%)
FastLAS	5,275.68	8	**Pickup Truck**: **3** (**37.50**%) [**Rubber**: **3** (**37.50**%)], Double Bus: 3 (37.50%), Minivan: 1 (12.50%), Biplane: 1 (12.50%)
Total	7,990.33	32	**Pickup Truck**: **27** (**84.38**%) [**Rubber**: **27** (**84.38**%)], Mountain Bike: 9 (28.13%), Double Bus: 3 (9.38%), Minivan: 1 (3.13%), Biplane: 1 (3.13%)

Table 19
Rule extraction results of the "sports motorcycle" class of the first iteration on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	83.12	4	**Dirtbike**: **4** (**100.00**%) [**North**: **3** (**75.00**%)]
FOLD-R++	2,436.15	20	**Dirtbike**: **20** (**100.00**%) [**North**: **8** (**40.00**%)]
FastLAS	13,933.60	42	**Dirtbike**: **33** (**78.57**%) [**North**: **33** (**78.57**%)], Double Bus: 4 (9.52%), Mountain Bike: 2 (4.76%), School Bus: 1 (2.38%), Chopper: 1 (2.38%), Articulated Bus: 1 (2.38%)
Total	16,452.87	66	**Dirtbike**: **57** (**86.36**%) [**North**: **44** (**66.67**%)], Double Bus: 4 (6.06%), Mountain Bike: 2 (3.03%), School Bus: 1 (1.52%), Chopper: 1 (1.52%), Articulated Bus: 1 (1.52%)

Table 20
Rule extraction results of the "urban bicycle" class of the first iteration on the *VT:$\mathcal{H}4$* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	112.26	8	**Utility Bike**: **8** (**100.00**%) [**North**: **3** (**37.50**%), **South**: **3** (**37.50**%)]
FOLD-R++	2,385.05	49	**Utility Bike**: **49** (**100.00**%) [**North**: **9** (**18.37**%), **South**: **9** (**18.37**%)], Private Jet: 3 (6.12%)
FastLAS	17,348.72	19	**Utility Bike**: **18** (**94.74**%) [**North**: **15** (**78.95**%), **South**: **14** (**73.68**%)], Road Bike: 1 (5.26%)
Total	19,846.03	76	**Utility Bike**: **75** (**98.68**%) [**North**: **27** (**35.53**%), **South**: **26** (**34.21**%)], Private Jet: 3 (3.95%), Road Bike: 1 (1.32%)

C.5. Primary Purpose: Hierarchy 1

C.5.1. Confusion Matrices

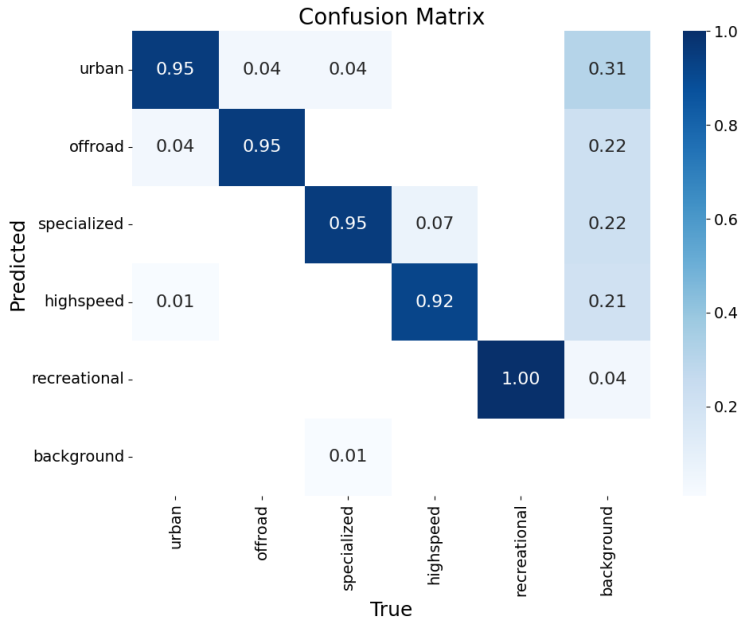

Fig. 31. Confusion matrix of the *PP:H1* model on the *Super-CLEVR* validation set after the initial model training.

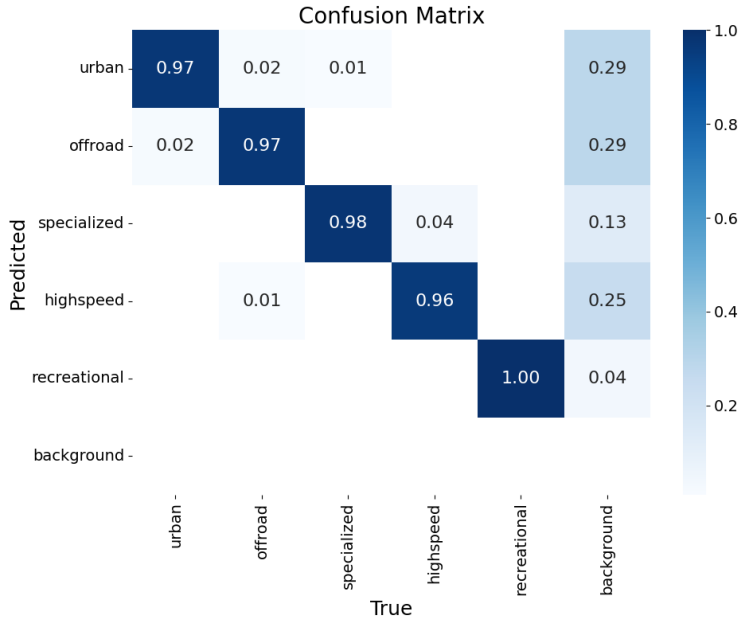

Fig. 32. Confusion matrix of the *PP:H1* model on the *Super-CLEVR* validation set after the model mending.

C.5.2. F1-Confidence Curves

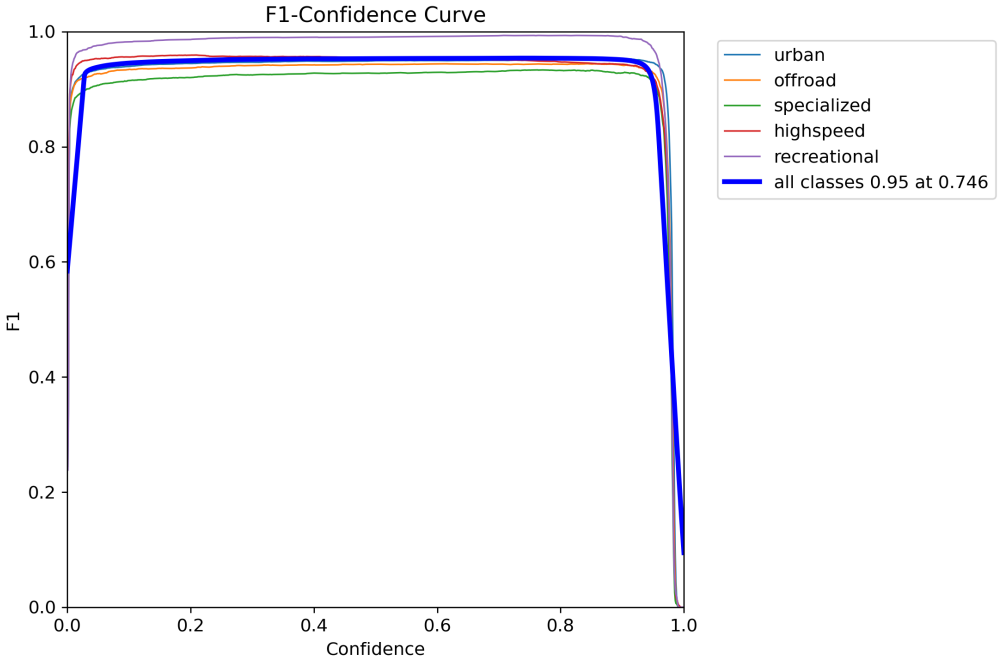

Fig. 33. F1-confidence curve of the *PP:$\mathcal{H}$1* model on the *Super-CLEVR* validation set after the initial model training.

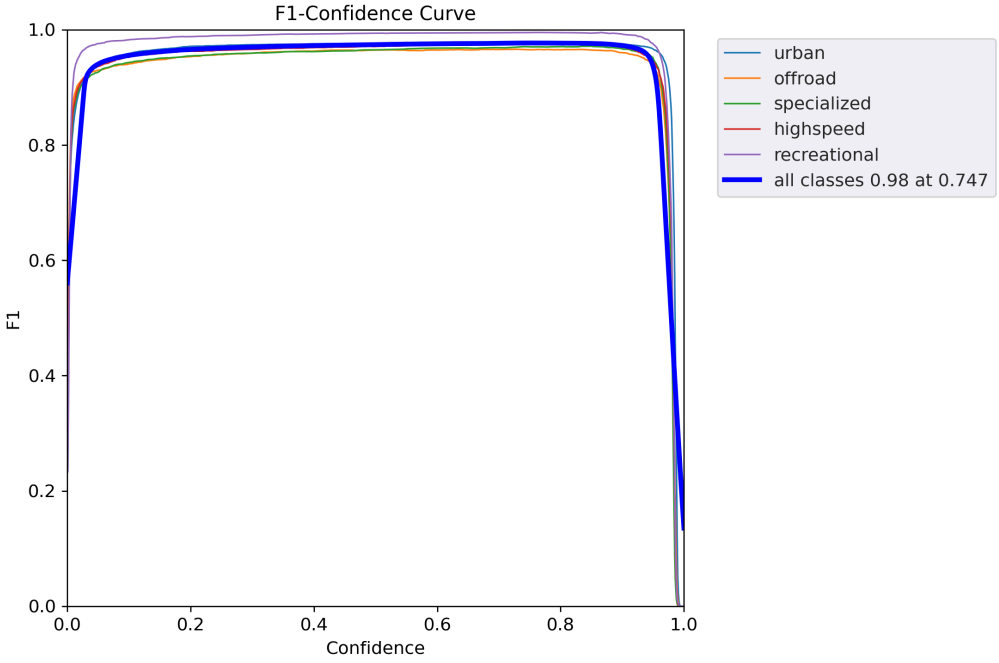

Fig. 34. F1-confidence curve of the *PP:$\mathcal{H}$1* model on the *Super-CLEVR* validation set after the model mending.

C.5.3. Model Training and Validation Performance Metrics

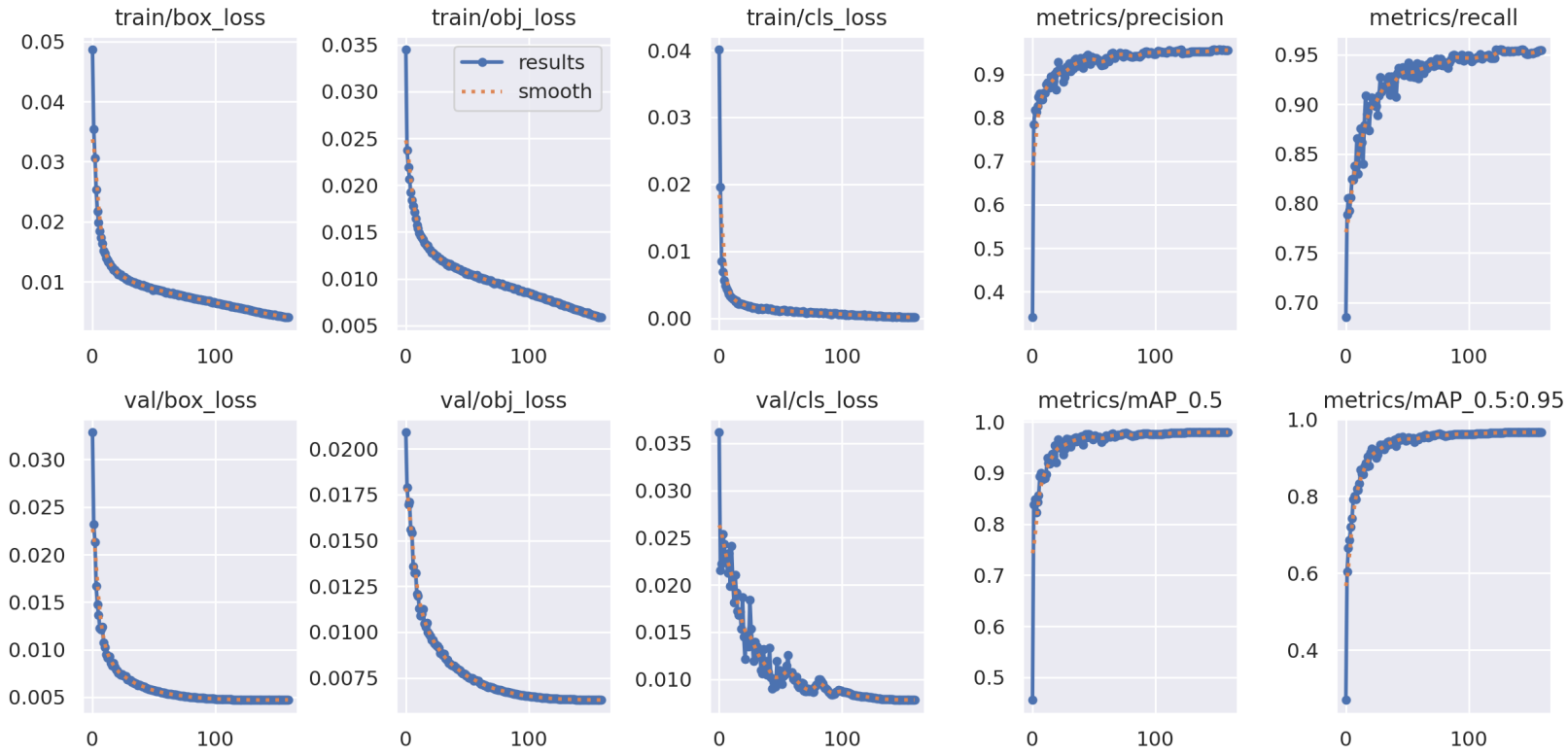

Fig. 35. *PP:$\mathcal{H}$1* model training and validation performance metrics on *Super-CLEVR* after the initial model training.

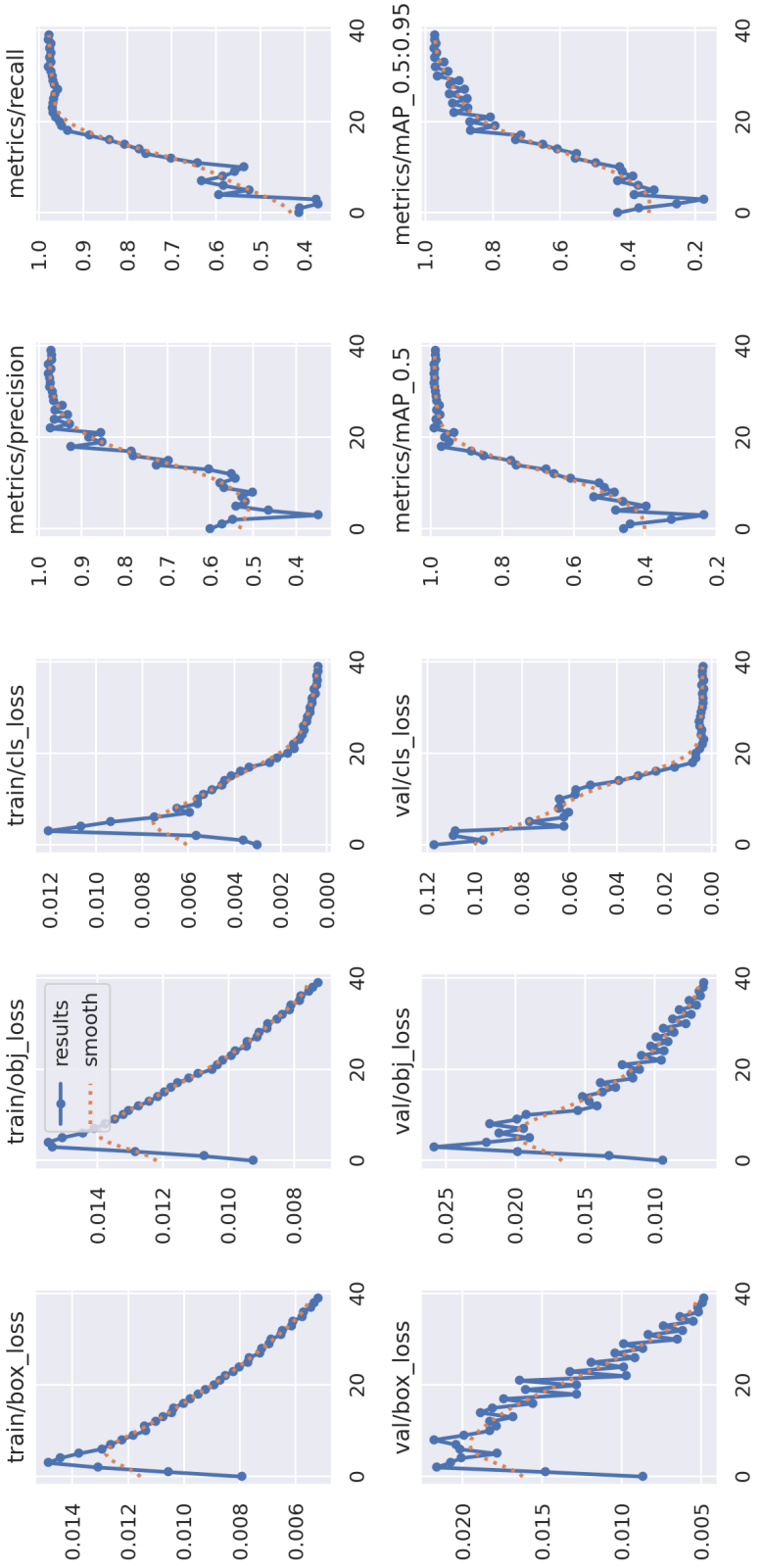

Fig. 36. *PP:H1* model training and validation performance metrics on *Super-CLEVR* after the model mending.

C.5.4. Rule Extraction and Selection Results

Table 21
Rule extraction results of the "offroad vehicle" class of the first iteration on the *PP:$\mathcal{H}$1* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	42.10	0	✗
FOLD-R++	1,911.14	27	**Pickup Truck**: **27** (**100.00**%) [**Rubber**: **27** (**100.00**%)], Utility Bike: 6 (22.22%)
FastLAS	10,464.07	10	**Pickup Truck**: **6** (**60.00**%) [**Rubber**: **5** (**50.00**%)], Chopper: 3 (30.00%), Biplane: 1 (10.00%)
Total	12,417.31	37	**Pickup Truck**: **33** (**89.19**%) [**Rubber**: **32** (**86.49**%)], Utility Bike: 6 (16.22%), Chopper: 3 (8.11%), Biplane: 1 (2.70%)

Table 22
Rule extraction results of the "specialized vehicle" class of the first iteration on the *PP:$\mathcal{H}$1* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	25.82	0	✗
FOLD-R++	6,398.83	27	**Articulated Bus**: **27** (**100.00**%) [**North**: **21** (**77.78**%)], Utility Bike: 6 (22.22%)
FastLAS	6,272.38	3	**Articulated Bus**: **2** (**66.67**%) [**North**: **1** (**33.33**%)], Road Bike: 1 (33.33%)
Total	12,697.03	30	**Articulated Bus**: **29** (**96.67**%) [**North**: **22** (**73.33**%)], Utility Bike: 6 (20.00%), Road Bike: 1 (3.33%)

Table 23
Rule extraction results of the "urban vehicle" class of the first iteration on the *PP:$\mathcal{H}$1* model for *Super-CLEVR*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	93.84	7	**Utility Bike**: **7** (**100.00**%) [**North**: **3** (**42.86**%), **South**: **3** (**42.86**%)], Road Bike: 1 (14.29%)
FOLD-R++	4,813.27	43	**Utility Bike**: **43** (**100.00**%) [**North**: **11** (**25.58**%), **South**: **14** (**32.56**%)], Fighter Jet: 1 (2.33%), Biplane: 1 (2.33%), School Bus: 1 (2.33%)
FastLAS	18,314.71	21	**Utility Bike**: **18** (**85.71**%) [**North**: **17** (**80.95**%), **South**: **13** (**61.90**%)], Fighter Jet: 1 (4.76%), Private Jet: 1 (4.76%), Sedan: 1 (4.76%)
Total	23,221.82	71	**Utility Bike**: **68** (**95.77**%) [**North**: **31** (**43.66**%), **South**: **30** (**42.25**%)], Fighter Jet: 2 (2.82%), Private Jet: 1 (1.41%), Sedan: 1 (1.41%), Biplane: 1 (1.41%), School Bus: 1 (1.41%)

Appendix D. ImageNet Experiments

D.1. Vehicle: Hierarchy 1

D.1.1. Experimental Results

Second Rule Extraction and Selection Iteration. Following the first model mending iteration, the overall performance of the model improved substantially. The four previously underperforming classes now all reported Top-1 accuracies greater than or equal to 90.00%, well above the target class threshold τ_c of 86.00%, indicating that the initial rare slices had been successfully resolved. However, we conducted a second iteration of the SDM pipeline to investigate any remaining deficiencies. Despite the overall improvement, the "van" class saw its performance drop to 85.75%, falling below our threshold τ_c and becoming the new performance bottleneck. We again employed our rule extraction module, tasking ILP systems to find rules that identify a potential rare slice within this newly problematic class. As in the previous iteration, we analysed the rules to identify underlying patterns and used the same hypothesis formation process and rare slice hypothesis threshold τ_h of 33.33%. The results of this second rule extraction iteration for the "van" class are summarised in Tables 27-29. The ILP systems showed a greater divergence in performance in this iteration. Both *Popper* and *FOLD-R++* failed to identify any relevant rules across all hyperparameter configurations. In terms of verbosity, their output was minimal; *Popper* extracted only one rule and *FOLD-R++* extracted 12, none of which were relevant to the underlying problem. In contrast, *FastLAS* was once again the most effective system. It was also the most verbose, generating a total of 64 rules, 40 of which agreed with our candidate hypothesis across almost all configuration settings. This suggests that the remaining performance issue was more subtle to identify. A detailed breakdown of the rules extracted for the underperforming "van" class is provided in Table 30. The evidence pointed to the "minivan" subclass as the primary source of the problem. In particular, while *Popper* and *FOLD-R++* did not contribute any relevant rules, 62.50% of the rules extracted from *FastLAS* concerned the "minivan" subclass. The total across all ILP systems showed that 51.95% of all generated rules contained the "minivan" subclass. We hypothesised that this problem resulted from the difficulty of the model in distinguishing the "minivan" vehicle from other visually similar subclasses (e.g., "jeep", "station wagon"), a confusion masked by the more severe initial data imbalance. Since the percentage of 51.95% exceeds τ_h, this led to the selection of the following candidate rule for the second model mending:

– Van *first* candidate rule:
```
hard(V0) :- contains(V0,V1), minivan(V1).
```

where, as before, `V1` denotes a vehicle in an image `V0`.

Second Model Mending Iteration. For the second mending iteration, we augmented the training data with 500 new images from *ImageNet* for the "minivan" subclass according to the selected rule. We then retrained the best-performing model from the first iteration (the one mended over 10 epochs) for an additional 10 and 20 epochs. To refine the model without risking degrading its performance for classes that still rely heavily on previous training, we employed fine-tuning with a very low initial learning rate of 1.0×10^{-8}. As detailed in Table 31, after both 10 and 20 epochs of fine-tuning, the model achieved a new best Top-1 accuracy of 90.91% on the validation set. However, a detailed error analysis shown in Table 31 revealed that this slight increase in overall accuracy did not correspond to an improvement in the problematic "van" class, which saw its accuracy slightly decrease to 85.50%. In contrast, the other classes maintained or slightly improved their high performance: "leisure vehicle" at 90.75%, "motorcycle" at 96.00%, "offroad vehicle" at 91.33%, and "passenger car" at 90.75%. We concluded that the remaining classification errors were likely not solvable by further data balancing alone and may require more fundamental changes to the model architecture or data collection strategy. Therefore, we terminated the mending process at this stage.

D.1.2. Rule Extraction and Selection Results

Table 24
Rule extraction results of the "motorcycle" class of the first iteration on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	5.52	0	✗
FOLD-R++	0.15	33	**Moped**: **29** (**87.88**%), Scooter: 2 (6.06%)
FastLAS	65.73	70	**Moped**: **53** (**75.71**%), Minivan: 8 (11.43%), Motorhome: 3 (4.29%), Convertible: 3 (4.29%)
Total	71.40	103	**Moped**: **82** (**79.61**%), Minivan: 8 (7.77%), Motorhome: 3 (2.91%), Convertible: 3 (2.91%), Scooter: 2 (1.94%)

Table 25
Rule extraction results of the "offroad vehicle" class of the first iteration on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	8.60	0	✗
FOLD-R++	0.15	6	Mountain Bike: 6 (100%)
FastLAS	115.17	94	**Jeep**: **61** (**64.89**%), Pickup Truck: 12 (12.77%), Station Wagon: 10 (10.64%), Minivan: 5 (5.32%), Mountain Bike: 2 (2.13%)
Total	123.92	100	**Jeep**: **61** (**61.00**%), Pickup Truck: 12 (12.00%), Station Wagon: 10 (10.00%), Mountain Bike: 8 (8.00%), Minivan: 5 (5.00%)

Table 26
Rule extraction results of the "passenger car" class of the first iteration on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	7.01	0	✗
FOLD-R++	0.13	20	**Station Wagon**: **14** (**70.00**%), Convertible: 2 (10.00%)
FastLAS	69.00	93	**Station Wagon**: **70** (**75.27**%), Convertible: 19 (20.43%), Moving Van: 2 (2.15%)
Total	76.14	113	**Station Wagon**: **84** (**74.34**%), Convertible: 21 (18.58%), Moving Van: 2 (1.77%)

Table 27
Rule extraction results of the second iteration of the *Popper* system on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Sample size		
	25%	50%	100%
VE:$\mathcal{H}$1			
V	✗ (0.54, 0, 0)	✗ (1.76, 0, 0)	✗ (3.64, 1, 0)

Table 28
Rule extraction results of the second iteration of the *FOLD-R++* system on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Sample size								
	25%			50%			100%		
	Exception ratio								
	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
VE:$\mathcal{H}$1									
V	✗ (0.01, 2, 0)	✗ (0.01, 2, 0)	✗ (0.01, 2, 0)	✗ (0.01, 0, 0)	✗ (0.01, 0, 0)	✗ (0.01, 0, 0)	✗ (0.02, 2, 0)	✗ (0.02, 2, 0)	✗ (0.02, 2, 0)

Table 29
Rule extraction results of the second iteration of the *FastLAS* system on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Sample size								
	25%			50%			100%		
	Rule head penalty								
	1	5	10	1	5	10	1	5	10
VE:$\mathcal{H}$1									
V	✓ (1.39, 6, 4)	✓ (1.21, 6, 4)	✗ (1.14, 0, 0)	✓ (3.21, 12, 7)	✓ (2.91, 10, 5)	✓ (2.67, 1, 1)	✓ (10.76, 14, 9)	✓ (10.41, 14, 9)	✓ (10.34, 1, 1)

Table 30
Rule extraction results of the "van" class of the second iteration on the *VE:$\mathcal{H}$1* model for *ImageNet*.

	Total runtime (s)	Total no. rules	Total no. rules per vehicle subclass
Popper	5.94	1	✗
FOLD-R++	0.12	12	✗
FastLAS	44.04	64	**Minivan**: **40** (**62.50**%), Moving Van: 17 (26.56%), Scooter: 4 (6.25%), Station Wagon: 2 (3.13%)
Total	50.10	77	**Minivan**: **40** (**51.95**%), Moving Van: 17 (22.08%), Scooter: 4 (5.19%), Station Wagon: 2 (2.60%)

Table 31
Top-1 accuracy of the *VE:$\mathcal{H}$1* model on the *ImageNet* validation set after the second model mending iteration. The left table shows the overall accuracy at different training epochs, the right table the accuracy for each of the five target classes.

Epochs	Top-1 acc. (%)
10	90.91
20	**90.91**

Target class	Top-1 acc. (%)
LV	90.75
M	96.00
OV	91.33
PC	90.75
V	**85.50**