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. The *slice discovery problem* involves detecting semantically meaningful slices on which the model performs poorly, called *rare slices*. We propose a modular neurosymbolic AI approach for slice discovery 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. Experiments conducted on datasets produced by our modified version of the *Super-CLEVR* data generator show that our approach can correctly identify rare slices and produce logical rules describing them. The rules can be fruitfully used to generate new training data to mend model behaviour and thus enhance its inference capabilities.[1]

Keywords: Neurosymbolic AI, Slice Discovery, Inductive Logic Programming

1. Introduction

Computer Vision (CV) [48] is a field of Artificial Intelligence (AI) that enables computer systems to extract valuable 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 [26], among which it is crucial to mention safety-critical applications, such as autonomous driving [53].

However, empirical studies, e.g. [40], 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 [16]. A *slice* is defined as a group of data sharing a set of attributes. For instance, regarding the task of identifying collapsed lungs in chest X-rays, it was observed in [37] that CV models base predictions on the presence of chest drains, which is a device used in treatment. Consequently, such models often make prediction errors on crucial slices where chest drains are absent, posing a significant risk of life-threatening false negatives.

Accurately detecting underperforming slices, called *rare slices*, allows one to carefully analyse such prediction errors and subsequently improve the model. However, identifying rare slices is a complex task, especially for high-dimensional data, e.g. images, where slices are very difficult to spot and extract.

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

Furthermore, it is non-trivial to understand what makes slices rare. In view of this, the *slice discovery problem* [16] 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 [20], 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* [31], a well-known synthetic dataset with a data generator for images of vehicles organised in hierarchical classes.

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 training and validation data generation, image classification, scene graph generation describing the semantic contents of images, rule learning to detect rare slices, and neural network model mending. To achieve these tasks, we provide a generator for datasets with rare slices leveraging *Super-CLEVR*, and on which we train *YOLOv5* [41]. 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++* [50], and *FastLAS* [29], 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. [16] considered the generation of rare slices in the context of the hierarchical class structure but did not take semantic relationships into account; this makes their method inapplicable to scenarios that 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.
3. We provide an implementation along with experimental results for datasets that we generate 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 considerably more detailed and extended related work section, (ii) the use of further ILP systems, (iii) a more rigorous experimental evaluation, 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 and ILP systems. Section 3 presents an introduction to the *Super-CLEVR* dataset and ILP. 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

We organise the review of related work by first presenting SDMs and then ILP systems.

2.1. Slice Discovery Methods

Several studies [5, 13, 24, 37] 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 [38] and fraud detection [23], 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 [22] 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 [42], 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 [19] 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 [16] exploits cross-modal embeddings and an error-aware Gaussian mixture model to discover and describe coherent slices, while the *Spotlight* method [12] 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 [47], 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 [43].

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 [35] 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 [18], 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 [39] 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 [46] is designed to test vision-and-language (ViL) models by discovering their slices from unlabelled image datasets. In [33], 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 [51], 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.

2.2. Inductive Logic Programming Systems

Previous research uses statistical or subsymbolic methods to address the slice discovery problem. We instead present a neurosymbolic approach consisting of a symbolic module based on ILP to extract human-

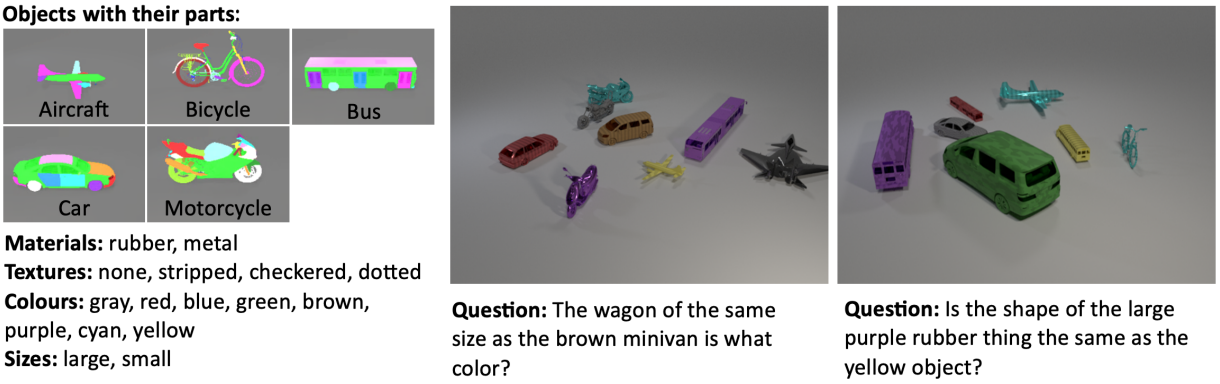

Fig. 1. The figure on the left shows images from *Super-CLEVR* [31] of vehicles made up of their parts characterised by four attributes, i.e. colour, size, material, and texture. The middle and right figures show examples of *Super-CLEVR* renderings with generated questions.

readable logical rules describing underperforming slices. ILP [9] is a rule-based learning method that aims to find patterns in data formulated through logical rules. It has found application in various fields, such as robotics [52], bioinformatics, for protein structure discovery [49], and medicine, for drug design [14, 17] and ECG waveform learning [25]. Further ILP applications can be found in [3, 27].

A well-known state-of-the-art ILP system is *Popper* [10], which implements the learning from failures approach by combining Answer Set Programming (ASP) [32] and Prolog [2]. Another modern ILP system is *ILASP* [28], which is designed for learning ASP programs containing normal rules, choice rules, and hard and weak constraints from examples. Furthermore, *FastLAS* [29] is a recent ILP system based on the context-dependent learning from answer sets framework used by *ILASP*. It comes with several restrictions, i.e. it is not as general as *ILASP*, but it is significantly more scalable and allows users to express their optimisation criteria. In [50], the authors present *FOLD-R++*, which is an improvement in terms of efficiency and scalability of the *FOLD-R* inductive learning algorithm [44] that learns ASP rules from mixed data for classification tasks.

Recently, some neurosymbolic ILP systems that attempt to combine the advantages of ILP with those of neural networks have been proposed. For example, δILP [15] is a differentiable ILP system for learning explicit human-readable symbolic rules, which is robust to noisy and ambiguous data and does not deteriorate when applied to unseen test data. αILP [45] instead is an extension of δILP that combines object-centric perception with ILP to learn representations of visual scenes as logic programs.

3. Preliminaries

In this section, we provide an overview of the *Super-CLEVR* dataset and briefly recall ILP.

3.1. Super-CLEVR

Inspired by the seminal work on *CLEVR* [21], 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 subclasses, which make the dataset *hierarchical*, a crucial characteristic for the realisation of our SDM. For example, the "motorcycle" class contains "chopper", "sportbike", "dirtbike", and "scooter" subclasses. 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 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.

3.2. Inductive Logic Programming

Inductive Logic Programming [36] is a subfield at the intersection of ML and KRR that focuses on learning logical descriptions from examples, utilising background knowledge (B) and sets of positive (E^+) and negative (E^-) 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. 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* implements the learning from failures approach by combining ASP and Prolog. 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++* is an improvement of the *FOLD-R* algorithm for learning 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* is a system designed to perform learning tasks in the context of ASP. A key feature of *FastLAS* is its capability of handling 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. 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.

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 [44], which is to consider misclassifications as positive examples and correct classifications as negative examples to obtain rules describing rare slices.

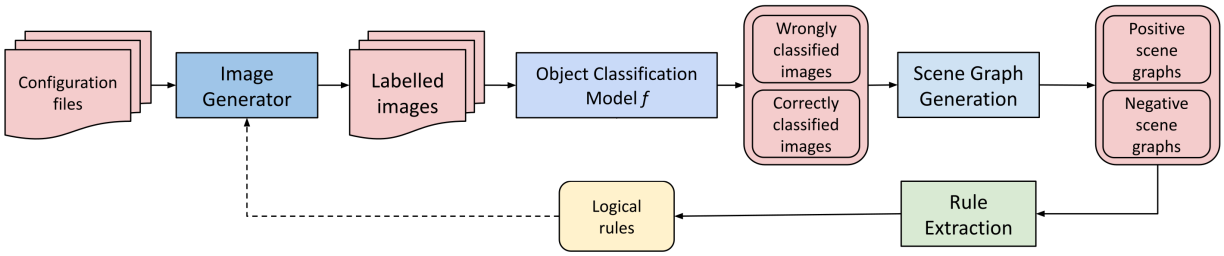

Fig. 2. Overview of the proposed neurosymbolic SDM architecture. The solid arrows show the data flow, while the dashed arrow represents the use of the extracted logical rules to generate further training data to improve classification performance.

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, synthetic datasets containing rare slices are generated 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.

4.1. Data Generation

The first step in the pipeline is concerned with data generation, i.e. producing datasets containing rare slices. To address this problem, we provide a methodology for generating datasets with rare slices, which will be detailed in Section 5.

At an abstract level, the task consists of creating a labelled dataset D, where elements are labelled with their attributes. In our *Super-CLEVR* setting, D consists of pairs (I, L), where I is an image and $L = \{(b_1, h_1), \ldots, (b_n, h_n)\}$ is its label, where each b_i is a bounding box of some object o_i in I, i.e. a tuple (x^-, y^-, x^+, y^+) where $(x^-, y^-), (x^+, y^+)$ are the corners of the bounding box, and h_i is in the underlying hierarchy $\mathcal{H}$ the root class of the subclass of o_i (e.g. the root class "motorcycle" for the vehicle subclass "dirtbike"), $1 \leqslant i \leqslant n$. Objects are identified by their bounding boxes, i.e. we can view b_i as an object ID.

To generate such datasets, we modified the *Super-CLEVR* data generator that renders images, such that we can control the distribution of objects depending on their class membership in $\mathcal{H}$ to create datasets that contain controlled rare slices. Each dataset is split into a training and validation set, where the respective information is provided in configuration files. Our data generator produces then a labelled dataset $D = \{(I_1, L_1), \ldots, (I_{N_s}, L_{N_s}\}$ with $N_s \approx n_s/\beta$ images per split s, where β is the average number of objects per image and n_s corresponds to the total number of objects per split.

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 efficacy assessment. Our modified *Super-CLEVR* generator offers the user flexibility to generate datasets with controlled rare slices and customised hierarchies, which allow one to study how a classification model reacts and the effectiveness of rule extraction to identify rare slices.

4.2. Object Detection and Classification

The dataset generated in the previous step is fed into the neural network model, and the classification results are collected. Specifically, the neural network model is expected to solve an object detection and classification problem: given an image I, it returns a set $f(I)$ of pairs $(\mathcal{B}, c)$, called *predictions*, where $\mathcal{B}$ is a

bounding box, which is identified with an object o, and c is a class from the root classes of the hierarchy $\mathcal{H}$; in case of *YOLOv5*, we also get for each object o the confidence cf in the bounding box and class detection.

The function f is obtained by training the neural network model on a dataset. For our aim, we use the training split of dataset D featuring rare slices from above, where we expect the model to underperform. After training, the model is run on the validation split of D and the validation images are then divided by the model output $f(I)$ into the sets E^+ and E^- of images that were incorrectly and correctly classified by f, respectively.

4.3. Scene Graph Generation

Scene Graph Generation (SGG) deals with generating a scene graph from an input image, which is a (labelled) directed graph $G = (V, E)$ with three types of nodes: object nodes, attribute nodes, and relation nodes. Each object in an image is located inside a bounding box and can have several attributes. Furthermore, different types of relations between pairs of objects, e.g. spatial relations like `behind`, may exist. Scene graphs provide a powerful semantic representation of images that can be exploited to understand visual data and spot their misclassification patterns. Different SGG methods and tools are available, mainly based on deep neural networks (see [30] for an overview).

In the realm of the *Super-CLEVR* setting, we can readily transform the sets E^+ and E^- of incorrectly resp. correctly classified images into suitable scene graphs G_{E^+} resp. G_{E^-}, by taking the bounding boxes (i.e. objects) along with the classes detected for them and supplementing their attributes. In that, we replace detected root classes with appropriate subclasses. The latter are not taken from the neural network model as it is not trained for such classification, but from the ground truth found in the dataset. Similarly, other object attributes, such as colour, size, material, and direction, are obtained from the ground truth. This allows us to construct correct scene graphs that are on the positive (G_{E^+}), negative (G_{E^-}) set when the neural network correctly, incorrectly classifies them, respectively.

4.4. Rule Extraction via Inductive Logic Programming

We specify a *rule extraction problem* instance by translating each scene graph G_{E^+} resp. G_{E^-} into a logical representation $\mathrm{ILP}(G_{E^+})$ resp. $\mathrm{ILP}(G_{E^-})$, and then assembling with them the sets E^+_{ILP} and E^-_{ILP} of positive and negative examples, respectively, and the background knowledge B describing semantic information about objects in the images. This is complemented with optional domain knowledge and auxiliary control information. Then, E^+_{ILP}, E^-_{ILP}, and B are fed into a rule extraction system. Notably, the positive examples E^+_{ILP} are 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 related mode bias used. 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 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 (again, the third set) combined with h that includes the `hard(32)` atom. Background knowledge B for each example is derived from the ground truth 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 in Fig. 3 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)`,

```
#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 background knowledge B in the language of *FastLAS*. The right side of the figure shows an excerpt of the related mode bias. The complete files defining the examples and mode bias can be found in the online repository.

`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`.

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 examples, one for background knowledge, and one for 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 line specifies the feature names of each column and the subsequent lines provide all examples. The complete encodings for the *FastLAS*, *Popper*, and *FOLD-R++* systems used in the experiments can be found in the online repository.

4.5. Model Mending

The extracted rules can be used to mitigate the influence of rare slices on model performance, leading to a more robust model. This improvement can be achieved by augmenting the training data with additional

images generated from the rules. To this end, rules may be selected by the user to meet preferences or desiderata; this may, to some degree, already be injected at the rule learning stage, depending on the expressiveness of the rule extraction system.

5. Rare Slice Generation Methodology

Motivated by the limitations of existing methods for generating rare slices in our context and the need for a reliable source of data to test our SDM, we present a taxonomy-based methodology to induce the occurrence of controlled rare slices in a model. For the application of this methodology, it is necessary to have a hierarchical dataset in which each class consists of a set of subclasses. Compared to related work [16], the novelty is that we propose a heuristic that bases its intuition on the fact that a CV model is more likely to confound visually similar objects that belong to different subclasses. Therefore, the heuristic is to define a taxonomy by separating similar subclasses into different target classes to induce the generation of rare slices in the classification model.

Super-CLEVR already defines a class hierarchy, but in our approach, the generator accepts as input also custom hierarchies. To generate rare slices, consider a hierarchy $\mathcal{H} = \{h_1 : \bar{s}_1, \ldots, h_m : \bar{s}_m\}$, where each h_i defines a root class and $\bar{s}_i$ is a set of subclasses for that root class. Then, consider a set $P = \{(c_1, c_2), \ldots, (c_{n-1}, c_n)\}, n \geqslant 1$, of pairs of visually similar subclasses, each belonging to some $\bar{s}_i$, that the CV model is likely to confound. Similar to [16], we construct a dataset D such that, for a given class $h_i \in \mathcal{H}$, one of its subclass c_i appearing in a tuple of P occurs with a very low probability α in D. We use the following methodology that we describe for the *Super-CLEVR* setting; applying it, with appropriate adjustments, in similar application settings is suggestive.

1. We create a slice configuration file containing a set $S = \{c_i \mid c_i \text{ appears in a tuple of } P\}$ of visually similar subclasses. 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. The subclasses may have further attributes that make the slices more specific such as picking a particular colour. The subclasses in S are defined as rare, while any other subclass is non-rare.
2. We associate with each rare subclass $c_i \in S$ a low probability α; lower α means a lower probability of creating a rare object.
3. A split configuration file is created defining the total number n_s of objects per split $s \in \{train, validation\}$ and whether the split should contain rare slices. We usually create rare slices in the training split but validate on splits without rare slices, i.e. with a uniform distribution of objects.
4. A configuration file is generated containing the specification of all rare slices and other subclasses, including their attributes, in the class hierarchy $\mathcal{H}$.
5. The configuration files are passed to the modified image generator, which computes for each c_i and s the total number $n_{c_i,s} = \frac{\alpha \cdot n_s}{100}$ of rare objects to be generated, rounded to an integer. If $n_{c_i,s} = 0$, the process stops with an error that no rare objects can be generated with the given n_s and asks the user to increase it. Otherwise, $n_{c_i,s}$ ($\geqslant 1$) rare objects are randomly distributed among the rendered images, where $n_s - \sum_{i=1}^{n} n_{c_i,s}$ is the number of random non-rare objects to be generated.

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.

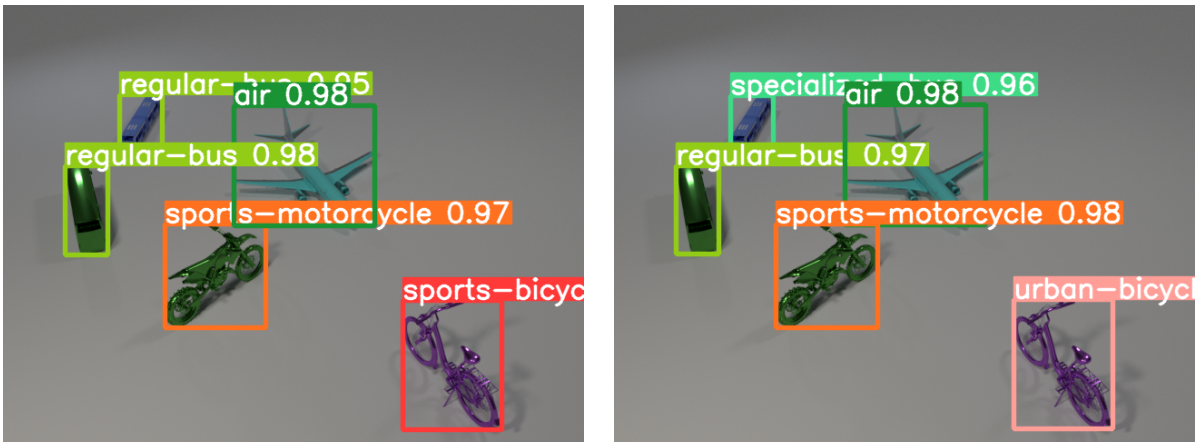

Fig. 4. The left figure shows a scene, based on hierarchy 4 of the *Vehicle Type* taxonomy, in which vehicles corresponding to the "utility bike" and "articulated bus" rare slices are misclassified by *YOLOv5* into the "sports bicycle" and "regular bus" classes, respectively. In contrast, the right figure shows the same scene in which such vehicles are correctly classified after model mending into their "urban bicycle" and "specialized bus" classes, respectively.

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. This section first describes the experimental setup and then presents an overview of the results obtained. All data and details are available in the online repository.

6.1. Experimental Setup

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), 1024GB of DRAM, four NVIDIA RTX A5000 GPUs (each having 24GB of VRAM), and the CUDA 12.2 API. In the following, we describe the experimental setup for each module of our SDM architecture.

Taxonomies. First, we considered the following 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, separating the vehicle subclasses of some pairs into different classes. In this way, we induced the generation of several rare slices to test the SDM implementation. The two taxonomies are listed and described below:

1. *Vehicle Type (VT)* classifies vehicles according to their type in a multilevel taxonomy, where classes become more and more specific at each level. The name of a class suggests the vehicles it contains. For example, the "air vehicles" class contains air vehicles such as "airliner" and "biplane". In contrast, the "scooter" and "mountain bike" vehicles are in the "land vehicles" class, but also in the one 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 vehicles, as shown in Fig. 6, where the vehicles are the leaves of the taxonomy. In the following, the different hierarchies considered in the experiments to evaluate and compare the influence of rare slices on classification performance are reported. For each hierarchy, the corresponding classes constitute the targets of the neural network model.
   - *Hierarchy 1:* "air vehicles" and "land vehicles".
   - *Hierarchy 2:* "air vehicles", "two-wheeler", and "multi-wheeler".

- *Hierarchy 3:* "air vehicles", "bicycle", "motorcycle", "bus", and "car". These are the original classes used in *Super-CLEVR*.
- *Hierarchy 4:* "air vehicles", "sports bicycle", "urban bicycle", "sports motorcycle", "urban motorcycle", "regular bus", "specialized bus", "offroad car", and "urban car".

Note that we purposely defined hierarchies 1 and 2 to not satisfy the heuristic criterion, i.e. no pair of vehicle subclasses was separated into different classes. On the contrary, for hierarchies 3 and 4, we defined the classes based on the heuristics. In particular, for hierarchy 3, we separated the pair ("dirtbike", "mountain bike"), while for hierarchy 4, we separated the pairs ("dirtbike", "mountain bike"), ("articulated bus", "transit bus"), ("utility bike", "mountain bike"), and ("pickup truck", "sedan"). In this way, we wanted to assess the effectiveness of the proposed heuristic in generating rare slices depending on their presence or absence in the respective hierarchies.

2. *Primary Purpose (PP)* is based on the classification of vehicles according to their primary usage, as shown in Fig. 7. For example, "scooter" is in the "urban vehicles" class, which contains vehicles intended for urban transportation, while "dirtbike" is in the "offroad vehicles" class. The only hierarchy, referred to as *hierarchy 1*, used in the experiments comprises the following classes: "urban vehicles", "offroad vehicles", "specialized vehicles", "high-speed vehicles", and "recreational vehicles". We defined these classes to separate the following pairs of vehicle subclasses: ("articulated bus", "transit bus"), ("utility bike", "mountain bike"), ("pickup truck", "sedan"), and ("private jet", "airliner").

Datasets. A training set consisting of 10,000 images was generated for the two taxonomies. Each image contains at least three vehicles but no more than six, and the vehicle attributes taken into account with their respective values are: "shapes" (e.g. "scooter"), "materials" (e.g. "metal"), "colors" (e.g. "gray"), "sizes" (e.g. "small"), and "directions" (e.g. "southwest"). For the training set, one vehicle subclass was selected for each of the five pairs mentioned above, i.e. "dirtbike", "articulated bus", "utility bike", "pickup truck", and "private jet", respectively. Then, their occurrence probability α in the dataset was set to 0.05%. Depending on the taxonomy employed, these vehicle subclasses are the candidate rare slices; the remaining vehicle subclasses are uniformly distributed. Lastly, a validation set of 2500 images is created, where all vehicle subclasses are uniformly distributed to evaluate the model behaviour.

Neural Network. For each hierarchy, a *YOLOv5* model was trained on the training set running 80, 160, and 320 epochs. Then, each model trained for 160 epochs was evaluated on the validation set, and the results were inspected.

Rule Extraction. For the rule extraction experiments, a timeout of 3600 seconds was set. For evaluation and comparison, three different ILP systems were employed on the underperforming classes of each hierarchy to extract the rules identifying their rare slices; an example is shown in Fig. 5. These systems were run for three sample sizes of the validation examples, i.e. 25%, 50%, and 100%, to study their accuracy and scalability as the amount of available data varied. The three ILP systems and their respective parameter configurations are described below. All values mentioned were chosen experimentally to test the capabilities and limitations of the systems.

1. *Popper* was used in the *noisy* mode, which allows it to learn the minimal description length program from noisy data.
2. *FOLD-R++* was run for three values of the *exception ratio* hyper-parameter, i.e. 0.25, 0.50, and 0.75. This parameter represents the threshold of the ratio of false-positive examples (i.e. exceptions) to true-positive examples that a rule can entail.
3. *FastLAS* was used in the *opl* mode, which runs the original algorithm (known as *FastLAS1*). Furthermore, to use *FastLAS* in the mode that supports noisy examples, for each example its *penalty* has been specified, which is the cost of violating it. The penalty values for positive and negative examples were set to 4 and 2, respectively. The maximum number of variables per rule was limited to 2 to narrow down the hypothesis space. Three penalties, i.e. 10, 20, and 30, were also tested for the scoring

```
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. 5. Rules extracted by *Popper* for the "urban bicycle" class in hierarchy 4 of the *Vehicle Type* taxonomy, for sample size 100%. The rules correctly detect the "utility bike" rare slice.

function that charges such penalty values for each extracted rule head, to observe how they affect the quality of the output result.

Rule selection. A set of rules was obtained for each experimental configuration based on the hierarchy, class, ILP system, and parameter values considered. From their analysis and interpretation, the candidate rules for slice identification were derived by selecting the vehicle attributes that occurred more often in the extracted rules. 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 in characterising positive examples, i.e. rare slices.

Model Mending. According to the candidate rules identifying a slice of a hierarchy, 12 new images containing the vehicle attributes described by those rules were generated to correct the model behaviour on the problematic class. To this end, for each hierarchy, the previously trained *YOLOv5* model (160 epochs version) was considered and further trained on the respective new images for 10 and 20 epochs. Finally, each model retrained for 10 epochs was evaluated on the same validation set to see how the model performance changed.

6.2. Experimental Results

This section provides the experimental results for rare slice generation, rule extraction, and model mending.

6.2.1. Rare Slice Generation

The proposed method successfully generated rare slices only for those hierarchies based on the heuristic that separates similar vehicle subclasses into different classes. Moreover, such rare slices negatively impacted the inference capabilities of the respective *YOLOv5* models across all epoch values considered, highlighting the persistence of rare slices even as the number of training rounds increases. From the confusion matrices obtained after model validation, e.g. as in Fig. 8 and 9, it can be observed that our taxonomy-based method was effective in inducing the presence of rare slices in the neural network models of such hierarchies. Specifically, as expected, no rare slices occurred in the two respective neural network models of hierarchies 1 and 2 of the *VT* taxonomy, as shown in Fig. 8. For hierarchy 3, only the "dirtbike" vehicle was found to be a rare slice for the "motorcycle" class, thus showing that the other vehicles with low probability did not affect model performance such that they were considered rare slices. In contrast, four rare slices, "dirtbike", "articulated bus", "utility bike", and "pickup truck", were created for hierarchy 4 for the "sports motorcycle", "specialized bus", "urban bicycle", and "offroad car" classes, respectively, as shown in Fig. 9. Finally, for hierarchy 1 of the *PP* taxonomy, "pickup truck" was found to be a rare slice for the "offroad vehicles" class, "articulated bus" for the "specialized vehicles" class, "utility bike" for the "urban vehicles" class, and "private jet" for the "high-speed vehicles" class.

6.2.2. Rule Extraction

We structure the presentation of experimental results for rule extraction according to taxonomy hierarchies. Tables 1-3 show the results of the three ILP systems for the neural network models trained with 160 epochs; the complete collection of all results can be found in the online repository.

VT Taxonomy: Hierarchy 3. All ILP systems identified the "dirtbike" slice in the "motorcycle" class. In particular, *Popper* succeeded for sample sizes 50% and 100%, but not for 25%. In contrast, *FOLD-R++*

detected such a slice for all parameter configurations. *FastLAS* was always successful for sample size 100%, but not for 25% with rule head penalties 20 and 30, and 50% with 30.

VT Taxonomy: Hierarchy 4. All ILP systems identified the "utility bike" slice in the "urban bicycle" class, except *FastLAS* for sample size 100% which always timed out. Examples of rules extracted from *Popper* and *FastLAS* detecting the "utility bike" slice are shown in Fig. 5 and Sec. 4.4, respectively. Also for the "dirtbike" slice in the "sports motorcycle" class all systems were successful, except *FOLD-R++* for sample size 25% which failed to extract any rules. As for the "pickup truck" slice in the "offroad car" class, both *FOLD-R++* and *FastLAS* managed to detect it while *Popper* did not. However, *FastLAS* experienced difficulties, only being able to identify it for sample sizes 50% and 100% with rule head penalty 10. On the other hand, *FOLD-R++* succeeded for sample sizes 25% and 100%, but not for 50%. Finally, all systems detected the "articulated bus" slice in the "specialized bus" class. In particular, *FOLD-R++* was successful for all parameter configurations. In contrast, *Popper* identified such a slice for sample sizes 50% and 100%, but not 25%. *FastLAS* was almost only successful with rule head penalty 10 for all sample sizes.

PP Taxonomy: Hierarchy 1. All ILP systems identified the "utility bike" slice in the "urban vehicles" class, except *FastLAS* for sample size 100% which always timed out. In contrast, the "pickup truck" and "articulated bus" slices in the "offroad vehicles" and "specialized vehicles" classes, respectively, proved difficult for both *Popper* and *FastLAS* to detect. In particular, *Popper* never succeeded, while *FastLAS* almost only with rule head penalty 10. Specifically, for the "offroad vehicles" class for sample sizes 50% and 100%, and for the "specialized vehicles" class for 25% and 50%. As for *FOLD-R++*, it was successful for both classes, except for "specialized vehicles" for sample size 50%. Finally, the "private jet" slice in the "high-speed vehicles" class was identified by all systems. In detail, *Popper* only succeeded for sample size 100%, but not for 25% and 50%. Instead, *FOLD-R++* detected such a slice for all parameter configurations. In contrast, *FastLAS* was successful for sample sizes 25% and 50% with rule head penalties 10 and 20, but for 100% it always timed out.

6.2.3. Model Mending

Candidate rules for slice identification were derived by analysing the extracted rules and selecting those vehicle attributes that occurred more often in the ILP hypotheses. For example, from the rules extracted for the "urban bicycle" class in hierarchy 4 of the *VT* taxonomy and shown in Fig. 5, the following candidate rules were derived:

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

These rules identify the rare slice containing utility bikes facing south, north, and southeast. Then, each *YOLOv5* model was retrained on new images generated according to the respective candidate rules. For example, the model for hierarchy 4 of the *VT* taxonomy was retrained on new images generated according to the candidate rules for the "sports motorcycle", "specialized bus", "urban bicycle", and "offroad car" classes. For each hierarchy, the confusion matrix obtained from model validation shows that the respective *YOLOv5* model has already been significantly corrected after 10 epochs, and further training for 10 epochs did not lead to improvement. In particular, for hierarchy 3 of the *VT* taxonomy, model correction increased the recall of the "motorcycle" class from 94% to 98%. For hierarchy 4 of the *VT* taxonomy, model mending worked really well, as shown in the confusion matrices in Fig. 9. Specifically, the recall values for the "urban bicycle" and "sports motorcycle" classes increased by around 11%, while for the "offroad car" and "specialized bus" classes they increased by an average of 6%. The "sports motorcycle", "offroad car", and "specialized bus" classes are now well-detected, while "urban bicycle" has improved significantly, as shown in the scene in Fig. 4. Finally, for hierarchy 1 of the *PP* taxonomy, the "urban vehicles" and "offroad vehicles" classes improved by 1%. More significant improvements are observed for the "specialized vehicles" and "high-speed vehicles" classes: the former increased from 95% to 98%, the latter from 92% to 98%.

Table 1

Summary of results of the *Popper* system for the neural network models trained with 160 epochs. ✓ stands for the successful production of rare slice rules and ✗ stands for wrong or no rules obtained. The system running time in seconds is given for each result.

	Sample Size		
	25%	50%	100%
Vehicle Type: Hierarchy 3			
Motorcycle	✗ (3)	✓ (14)	✓ (30)
Vehicle Type: Hierarchy 4			
Urban Bicycle	✓ (12)	✓ (30)	✓ (72)
Sports Motorcycle	✓ (8)	✓ (27)	✓ (50)
Offroad Car	✗ (2)	✗ (8)	✗ (20)
Specialized Bus	✗ (3)	✓ (9)	✓ (30)
Primary Purpose: Hierarchy 1			
Urban Vehicles	✓ (12)	✓ (26)	✓ (57)
Offroad Vehicles	✗ (4)	✗ (8)	✗ (31)
Specialized Vehicles	✗ (2)	✗ (7)	✗ (18)
High-Speed Vehicles	✗ (8)	✗ (18)	✓ (47)

Table 2

Summary of results of the *FOLD-R++* system for the neural network models trained with 160 epochs. ✓ stands for the successful production of rare slice rules and ✗ stands for wrong or no rules obtained. The system running time in seconds is given for each result.

	Sample Size								
	25%			50%			100%		
	Ratio								
	0.25	0.50	0.75	0.25	0.50	0.75	0.25	0.50	0.75
Vehicle Type: Hierarchy 3									
Motorcycle	✓ (34)	✓ (34)	✓ (34)	✓ (73)	✓ (73)	✓ (73)	✓ (279)	✓ (277)	✓ (172)
Vehicle Type: Hierarchy 4									
Urban Bicycle	✓ (34)	✓ (35)	✓ (35)	✓ (230)	✓ (232)	✓ (118)	✓ (642)	✓ (569)	✓ (447)
Sports Motorcycle	✗ (6)	✗ (6)	✗ (6)	✓ (148)	✓ (198)	✓ (197)	✓ (781)	✓ (508)	✓ (513)
Offroad Car	✓ (54)	✓ (54)	✓ (54)	✗ (16)	✗ (17)	✗ (17)	✓ (794)	✓ (802)	✓ (799)
Specialized Bus	✓ (85)	✓ (158)	✓ (120)	✓ (96)	✓ (97)	✓ (176)	✓ (2003)	✓ (781)	✓ (722)
Primary Purpose: Hierarchy 1									
Urban Vehicles	✓ (132)	✓ (126)	✓ (127)	✓ (358)	✓ (267)	✓ (260)	✓ (2361)	✓ (545)	✓ (548)
Offroad Vehicles	✓ (102)	✓ (101)	✓ (102)	✓ (258)	✓ (260)	✓ (261)	✓ (242)	✓ (244)	✓ (241)
Specialized Vehicles	✓ (149)	✓ (148)	✓ (149)	✗ (17)	✗ (17)	✗ (17)	✓ (1938)	✓ (1935)	✓ (1930)
High-Speed Vehicles	✓ (264)	✓ (262)	✓ (110)	✓ (438)	✓ (431)	✓ (434)	✓ (703)	✓ (583)	✓ (546)

7. Discussion

From the confusion matrices, we observed that our taxonomy-based dataset generation method effectively induced the presence of rare slices in neural network models. Each rare slice was identified by at least two ILP systems in some specific configuration. Notably, both *FOLD-R++* and *FastLAS* succeeded on all slices, thus proving consistency. In contrast, *Popper* failed on the "pickup truck" and "articulated bus" slices, which also proved to be the most problematic slices for the other two systems. In this context, the results

Table 3

Summary of results of the *FastLAS* system for the neural network models trained with 160 epochs. ✓ stands for the successful production of rare slice rules, ✗ stands for wrong or no rules obtained, and T stands for timeout. The system running time in seconds is given for each result.

	Sample Size								
	25%			50%			100%		
	Rule head penalty								
	10	20	30	10	20	30	10	20	30
Vehicle Type: Hierarchy 3									
Motorcycle	✓ (165)	✗ (169)	✗ (171)	✓ (711)	✓ (704)	✗ (710)	✓ (1980)	✓ (2001)	✓ (1980)
Vehicle Type: Hierarchy 4									
Urban Bicycle	✓ (429)	✓ (430)	✓ (433)	✓ (1778)	✓ (1733)	✓ (1748)	T	T	T
Sports Motorcycle	✓ (268)	✓ (279)	✓ (277)	✓ (1243)	✓ (1233)	✓ (1188)	✓ (3186)	✓ (3146)	✓ (3118)
Offroad Car	✗ (131)	✗ (135)	✗ (136)	✓ (388)	✗ (391)	✗ (398)	✓ (1238)	✗ (1244)	✗ (1220)
Specialized Bus	✓ (184)	✗ (184)	✗ (176)	✓ (507)	✗ (514)	✗ (514)	✓ (1805)	✓ (1760)	✗ (1788)
Primary Purpose: Hierarchy 1									
Urban Vehicles	✓ (588)	✓ (581)	✓ (575)	✓ (1903)	✓ (1931)	✓ (1940)	T	T	T
Offroad Vehicles	✗ (188)	✗ (186)	✗ (191)	✓ (723)	✗ (722)	✗ (713)	✓ (2588)	✓ (2572)	✗ (2585)
Specialized Vehicles	✓ (133)	✗ (139)	✗ (135)	✓ (403)	✗ (416)	✗ (403)	✗ (1551)	✗ (1558)	✗ (1539)
High-Speed Vehicles	✓ (416)	✓ (423)	✗ (414)	✓ (1194)	✓ (1177)	✗ (1190)	T	T	T

suggest that ILP systems tend to be more likely to succeed as the amount of available examples increases. However, it is interesting and important to note that both *FOLD-R++* and *FastLAS* identified most of the rare slices even for sample size 25%. This was not the case for *Popper*, which halved the number of identified slices by going from 100% to 25% sample size, thus showing less robustness. Interestingly and unexpectedly, *FOLD-R++* failed on the "sports motorcycle" class for sample size 25% and on the "offroad car" and "specialized vehicles" classes for sample sizes 50%, as shown in Table 2. The reasons behind this need further investigation. However, it can be speculated that the reason depends on the characteristics and distribution of the positive and negative examples considered.

In addition to requiring less data for slice discovery, smaller samples have the advantage of significantly reducing the running time of ILP systems, as shown in the Tables 1-3. In particular, using smaller samples was necessary for *FastLAS*, which turned out to be the slowest system, timing out for three classes with sample size 100%. One possible reason why *FastLAS* is slower may be its penalty mechanism and scoring function, which make the optimisation problem more challenging to solve. In contrast, *FOLD-R++* and *Popper* never ran into timeout, with the latter being the fastest system, probably because *Popper* lacks negation in the extracted rules. Indeed, it is important to consider that both *FOLD-R++* and *FastLAS* provide rules with negation, which greatly widens the hypothesis space. On the other hand, 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. Finally, the values of *exception ratio* and *rule head penalty* subtly influenced the results and running times. In particular, *exception ratio* had a substantial impact on the *FOLD-R++* running times of some classes, e.g. in the "urban vehicles" class for hierarchy 1 of the *PP* taxonomy, as shown in Table 2. In contrast, *rule head penalty* influenced the results of slice identification by *FastLAS*.

Model mending based on the rules extracted from our SDM proved effective for all hierarchies. In particular, the confusion matrices showed significant improvement in the recall values of the classes affected by rare slices after re-training the classification models on the new rule-based images. The differences in the improvements in recall values suggest that the effectiveness of model mending varies by class. Finally, the results indicate that model mending allows significant improvements without extensive re-training, as the recall values plateaued after 10 epochs.

8. Conclusion and Future Work

In this work, we have presented a neurosymbolic AI approach to address the slice discovery problem. In particular, we have provided a modular architecture and an implementation that leverages the *Super-CLEVR* dataset and its synthetic data generator. The architecture connects data generation, model classification, and rule extraction from scene graphs via ILP to detect rare slices that are misclassified. Furthermore, we have described a taxonomy-based methodology for generating datasets in which rare slices occur. The experiments have shown that the datasets with rare slices could be generated reliably, and that the rule extraction approach using ILP could produce useful rules describing rare slices. Further training the neural network model with data generated from these rules has resulted in significant performance improvement.

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 aims in different directions. One is to improve the current implementation and make it more user-friendly and accessible. Another direction is the generation of native scene graphs, for which tools and methods, as mentioned in Section 4, will have to be examined and assessed; generating well-suited scene graphs by them is an interesting research issue. Then, the plan is to extend the work to consider relationships between objects, which would be an important added value in identifying rare slices with further precision but at the expense of higher computational cost. Moreover, exploring further ILP systems as well as other rule learning approaches, such as those provided by Statistical Relational Learning, e.g. LERND [34], is on our agenda.

References

[1] V. Boreiko, M. Hein and J.H. Metzen, Identifying Systematic Errors in Object Detectors with the SCROD Pipeline, in: *Proceedings of the 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, Lecture Notes in Computer Science, 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] 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.

[13] 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.

[14] 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, Lecture Notes in Computer Science, Vol. 2838, Springer, 2003, pp. 156–167. doi:10.1007/978-3-540-39804-2_16.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] 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.

[21] 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.

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

[23] 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.

[24] 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.

[25] 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, Lecture Notes in Computer Science, Vol. 1211, Springer, 1997, pp. 126–129. doi:10.1007/BFB0029443. https://doi.org/10.1007/BFb0029443.

[26] 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.

[27] 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.

[28] M. Law, A. Russo and K. Broda, The ILASP system for learning Answer Set Programs, 2015.

[29] 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, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 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.

[30] 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.

[31] 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.

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

[33] 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, Lecture Notes in Computer Science, Vol. 15091, Springer, 2024, pp. 235–252. doi:10.1007/978-3-031-73414-4_14.

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

[35] 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.

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

[37] 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.

[38] 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, Lecture Notes in Computer Science, Vol. 15198, Springer, 2024, pp. 3–13. doi:10.1007/978-3-031-72787-0_1.

[39] 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: *Proceedings of the 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.

[40] B. Recht, R. Roelofs, L. Schmidt and V. Shankar, Do ImageNet Classifiers Generalize to ImageNet?, in: *Proceedings of the 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.

[41] 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.

[42] 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.

[43] 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.

[44] 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.

[45] 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.

[46] 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.

[47] 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.

[48] 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.

[49] 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., Lecture Notes in Computer Science, Vol. 1446, Springer, 1998, pp. 53–64. doi:10.1007/BFB0027310. https://doi.org/10.1007/BFb0027310.

[50] 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, Lecture Notes in Computer Science, Vol. 13215, Springer, 2022, pp. 224–242. doi:10.1007/978-3-030-99461-7_13.

[51] 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.

[52] 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, Lecture Notes in Computer Science, Vol. 14363, Springer, 2023, pp. 154–165. doi:10.1007/978-3-031-49299-0_11.

[53] 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: *Proceedings of the 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.

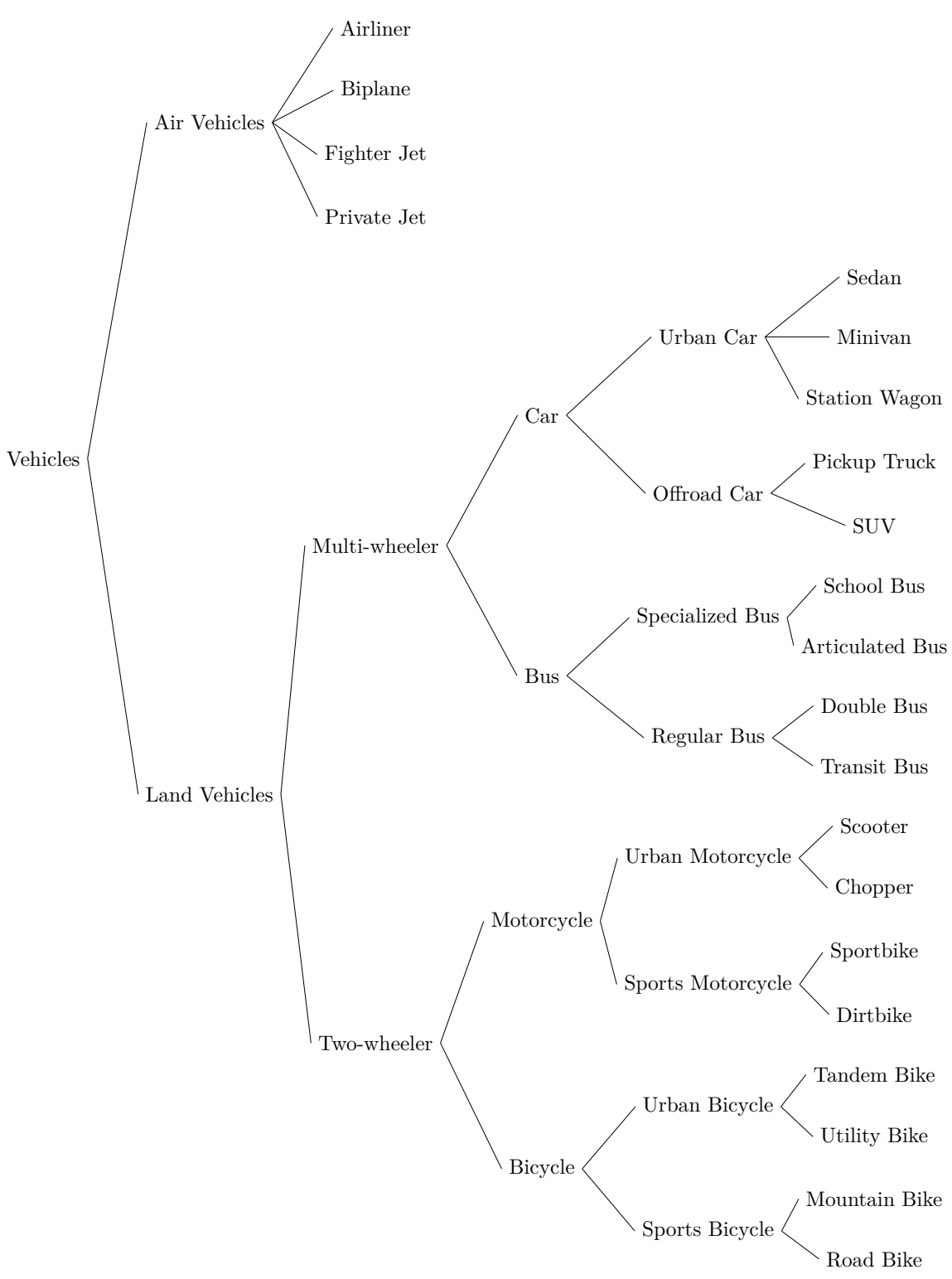

Fig. 6. *Super-CLEVR Vehicle Type* taxonomy.

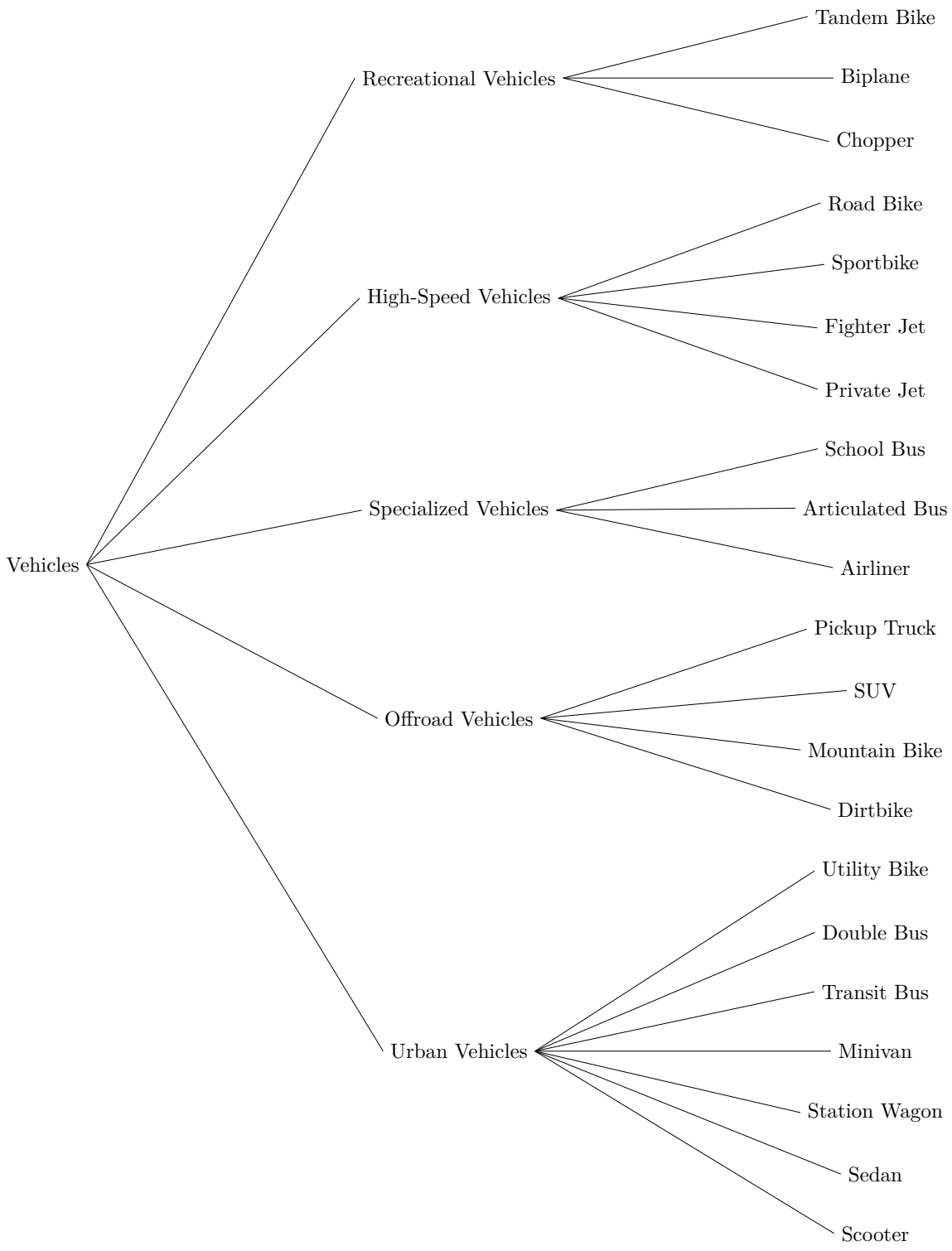

Fig. 7. *Super-CLEVR Primary Purpose* taxonomy.

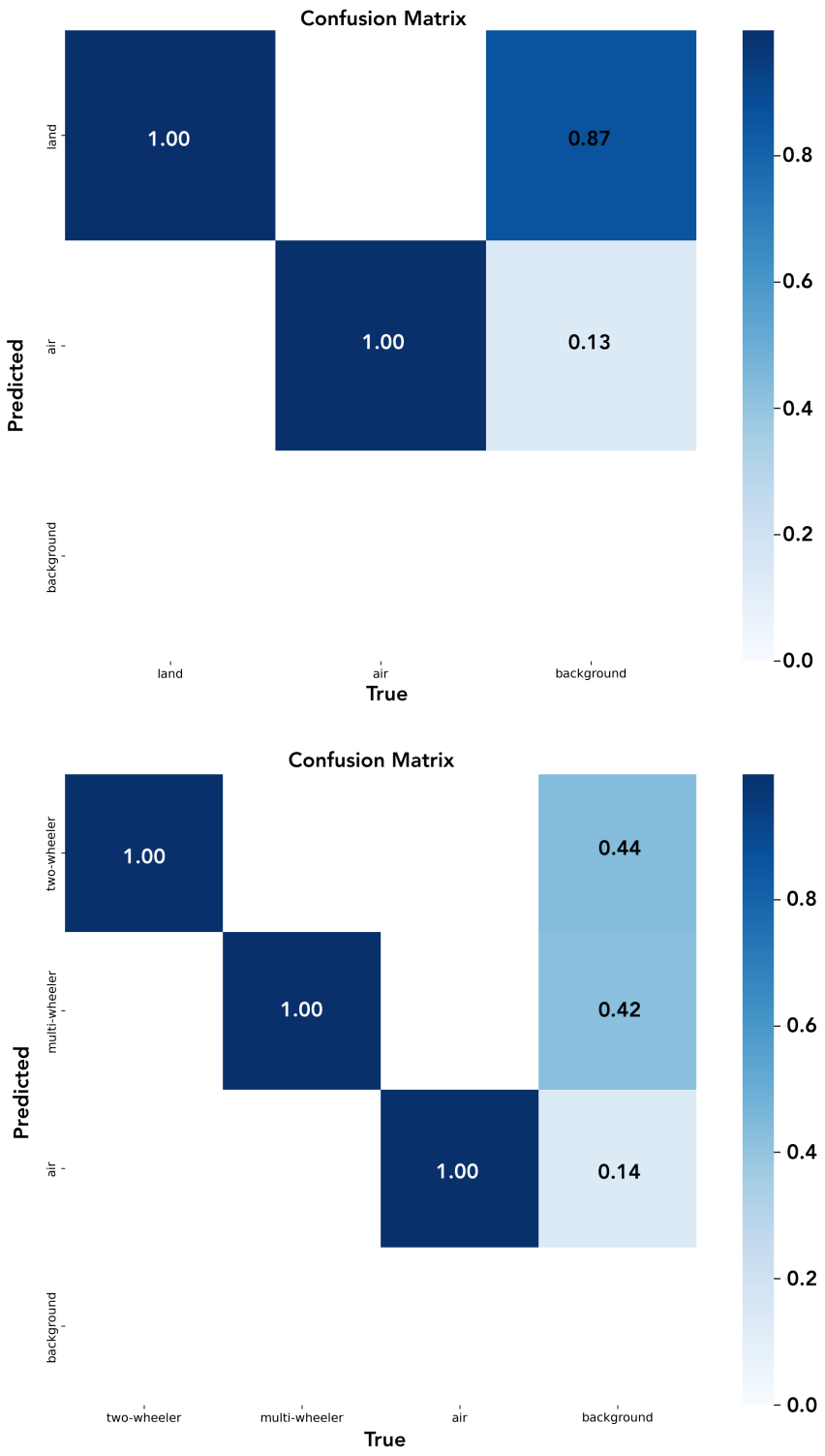

Fig. 8. The top figure refers to hierarchy 1 of the *Vehicle Type* taxonomy, while the bottom figure refers to hierarchy 2. Both figures show the respective confusion matrix from model validation after training on the dataset with rare slices. Note that rare slices did not affect the classification performance of both neural network models.

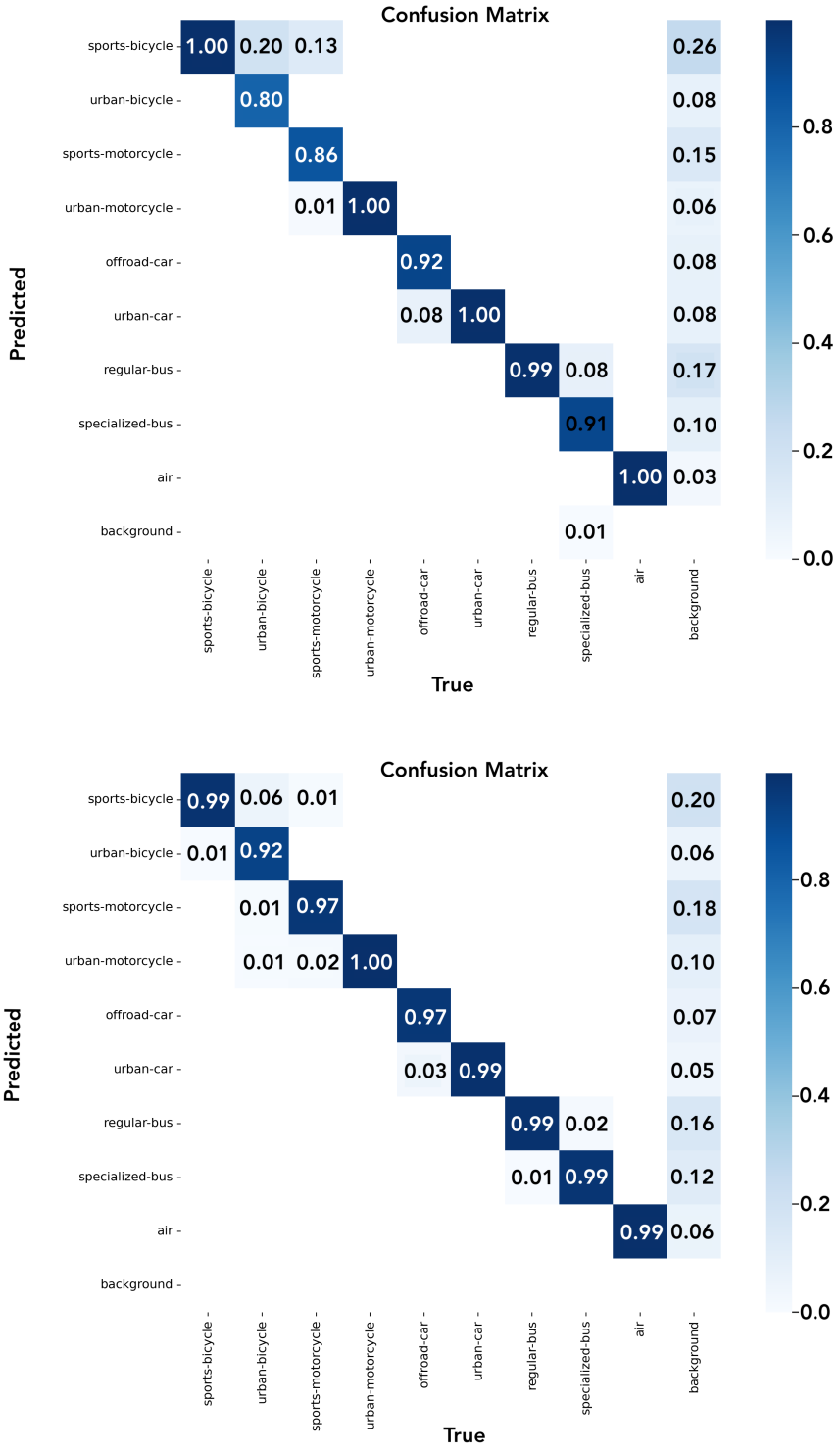

Fig. 9. Both figures refer to hierarchy 4 of the *Vehicle Type* taxonomy. The top figure shows the confusion matrix from model validation after training on the dataset with rare slices. The bottom figure shows the improvement after model mending for those classes where rare slices were detected.