Research Square

Preprints are preliminary reports that have not undergone peer review.
They should not be considered conclusive, used to inform clinical practice,
or referenced by the media as validated information.

Chebifier 2: An Ensemble for Chemistry

Simon Flügel

simon.fluegel@uos.de

Osnabrück University

Martin Glauer

Otto-von-Guericke University Magdeburg

Aditya Ganesh Khedekar

Otto-von-Guericke University Magdeburg

Fabian Neuhaus

Otto-von-Guericke University Magdeburg

Charlotte Tumescheit

Idiap Research Institute

Janna Hastings

Idiap Research Institute

Till Mossakowski

Osnabrück University

Research Article

Keywords: ChEBI, multi-label classification, hierarchical multi-label classification, ensemble learning, heterogeneous ensemble

Posted Date: November 6th, 2026

DOI: https://doi.org/10.21203/rs.3.rs-9023090/v2

License: This work is licensed under a Creative Commons Attribution 4.0 International License.
Read Full License

Additional Declarations: The authors declare no competing interests.

Chebifier 2: An Ensemble for Chemistry

Simon Flügel *[0000-0003-3754-9016][1], Martin Glauer [0000-0001-6772-1943][2], Aditya Ganesh Khedekar [0009-0003-5454-6105][1,2], Fabian Neuhaus [0000-0002-1058-3102][2], Charlotte Tumescheit [0000-0002-7563-5575][3,4], Janna Hastings [0000-0002-3469-4923][3,4], Till Mossakowski [0000-0002-8938-5204][1]

[1*]Institute for Computer Science, University of Osnabrück, Neuer Graben 29, 49074 Osnabrück, Germany.
[2]Institute for Cooperating Systems, Otto von Guericke University Magdeburg, Universitätsplatz 2, 39106 Magdeburg, Germany.
[3]Idiap Research Institute, Rue Marconi 19, 1920 Martigny, Switzerland.
[4]Swiss Institute of Bioinformatics, Lausanne, Switzerland.

Contributing authors: simon.fluegel@uni-osnabrueck.de;

Abstract

Accurately classifying chemical structures is a central task in cheminformatics. Different approaches to classification exist, but they all come with drawbacks. Manual classification is time-intensive and hard to scale for large databases. Rule-based methods only work for some classes and require a high level of maintenance, while deep learning models lack reliability and explainability. In this work, we present Chebifier 2, an ensemble learning framework that combines deep learning and symbolic approaches to classify chemicals into the Chemical Entities of Biological Interest (ChEBI) ontology, covering a total of 2,234 classes.
The framework consists of two stages: an aggregation stage, in which base learner predictions are combined per class, and an inconsistency resolution stage, in which predictions are aligned with the axiomatisation of ChEBI. For the aggregation stage, this work evaluates five methods: Majority Voting, two Weighted Majority Voting schemes, and the meta-learning methods Learning To Rank and Dynamic Ensemble Selection. For the inconsistency resolution stage, three methods are evaluated: Hierarchy and Exclusion graphs, Iterative Local Refinement and a novel 3-step inconsistency resolution algorithm. The results show that ensemble learning can beat the best non-ensemble method by 5.9 percentage points in the macro F1 score. Particularly on small and chemically difficult

*Corresponding author

classes, the ensemble outperforms the baseline by a substantial margin ($\geq$ **10** percentage points on classes with fewer than 30 members).

Scientific Contribution

This work introduces Chebifier 2, an ensemble framework for the classification of chemical structures in the ChEBI ontology. It covers a larger part of ChEBI than previous approaches and outperforms existing methods, especially on small classes. Besides adapting existing aggregation and inconsistency resolution methods to ChEBI, this work also proposes a novel 3-step resolution method specialised in disjointness and subsumption relations.

Keywords: ChEBI, multi-label classification, hierarchical multi-label classification, ensemble learning, heterogeneous ensemble

1 Introduction

Over the past decades, the amount of knowledge available in the life sciences has grown exponentially. Making use of this knowledge requires organisation and the connection of new findings to existing knowledge. For example, the Chemical Entities of Biological Interest (ChEBI) ontology [1, 2] provides a manually curated classification hierarchy for chemicals with 238,130 entries (as of August 2026). While manual curation allows ChEBI to enforce a high level of quality, it cannot keep up with other chemistry databases such as PubChem [3], which contains 124 million compounds (as of August 2026) and grows by around 200,000 entries every month.

Therefore, there is an urgent need for an automated extension of ChEBI, i.e., the automatic annotation of new molecules with classes from the ChEBI ontology (cf. Figure 1). So far, different methods have been proposed for this task. Rule-based methods [4–9] are successful when applied to specific classes. Since the rules are developed by chemical experts, they can achieve a level of accuracy similar to that of human curators and even reduce errors. ClassyFire [4] in particular has become a tool that is used to assist ChEBI development. However, creating rules still requires a significant amount of human labour – in most cases, it is not possible to trivially extend a rule-based approach to new classes. In addition, not all classes in ChEBI are defined by structural features that can easily be captured with rules that are based on pattern matching against the chemical structure or easily computable chemical properties.

An extension of this approach has been suggested in [10], using a large language model to generate Python programs that classify chemicals. This approach shares some of the advantages of the manually crafted rules: the classification process is transparent and can provide natural-language explanations for each classification. However, without human oversight, the programs cannot make the same claims regarding correctness.

An alternative approach is provided by machine learning [11–13], which does not require hand-crafted rules, but instead learns directly from data. One advantage of machine learning methods is that existing models may be retrained on new classes

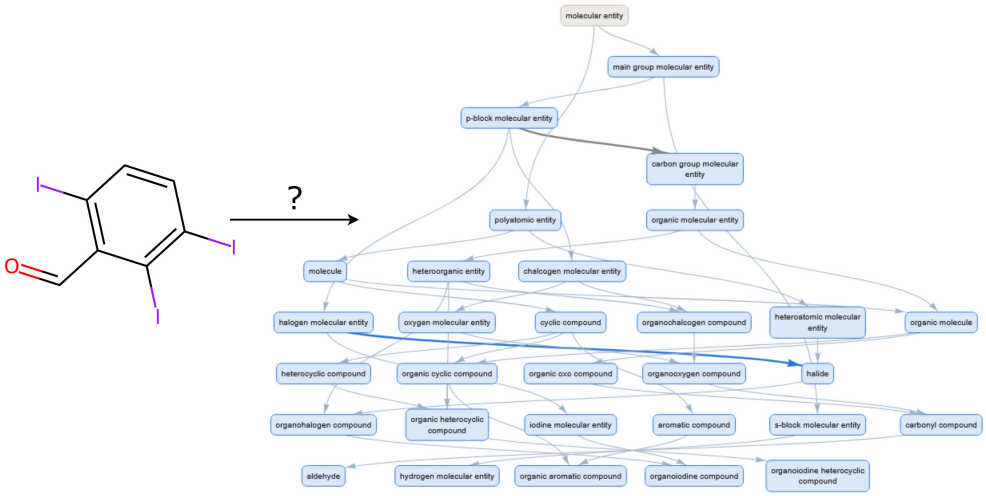


Fig. 1: Example of the ChEBI classification task for a molecule that is not yet part of the ontology: *2,3,6-Triiodobenzaldehyde*. The goal is to assign ChEBI classes to novel molecules in a manner consistent with the ontology's structure. This work proposes to replace the question mark not with a single method, but with an ensemble of methods and an inconsistency resolution mechanism. The right-hand side shows the ChEBI classes which Chebifier 2 would assign to *2,3,6-Triiodobenzaldehyde*.

with minimal additional effort. However, the data dependency of machine learning approaches can also be a downside. Deep learning models in particular fail to learn complex chemical classes if insufficient training data is available. Another disadvantage is that they lack interpretability: deep learning models are inherently black boxes. While efforts towards a more human-understandable classification have been made [13, 14], this remains an open research topic.

This work proposes to combine the different classification methods with a heterogeneous ensemble. The ensemble has to address several challenges that are particular to the ChEBI extension task and the integrated methods: Firstly, the models cover different (but often overlapping) parts of ChEBI which the ensemble needs to integrate into a single prediction. Secondly, the ensemble needs to adhere to the ChEBI axiomatisation. The classes in the ontology are connected via subsumption and disjointness relations that must be respected. For example, if the ensemble classifies a molecule as both organic and inorganic, the prediction will be unusable. Based on these requirements, we have developed Chebifier 2, an ensemble framework that separates the prediction into two stages: the *ensemble aggregation stage* assigns weights and normalises the predictions for each class, compensating for missing predictions if necessary. The *inconsistency resolution stage* algorithmically fixes inconsistencies between ChEBI classes.[1]

For both stages, this work evaluates different methodologies: in ensemble aggregation, majority voting and meta-learning methods are used. Inconsistency resolution is

[1] A preliminary version of this architecture has been published in [15].

conducted with Hierarchy and Exclusion (HEX) graphs [16], Iterative Local Refinement [17] and a novel 3-step algorithm that exploits the structure of ChEBI. Chebifier 2 is available as a Python library [18] and via a web interface [19].

In the following sections we will discuss related work (Section 2), introduce the ensemble framework and the individual aggregation and resolution methods (Section 3), the application to ChEBI and the base learners (Section 4), the evaluation (Section 5) and finally, a conclusion (Section 6).

2 Related Work

In this work, we develop an ensemble architecture for a hierarchical multi-label classification problem. Two central aspects of this task are the aggregation mechanism, i.e., "What is the optimal way to aggregate base learner predictions into a single ensemble prediction?", and inconsistency resolution, i.e., "How do I make sure that the ensemble prediction is consistent with label constraints?".

2.1 Ensemble Aggregation

Moyano et al. [20] have divided ensemble aggregation techniques for multi-label problems into four categories: Binary Relevance combines a set of binary classifiers, one for each label, into a multi-label prediction. Methods such as Classifier Chains [21], Hierarchy of Multi-label Classifiers [22] and AdaBoost.MH [23] are built on this idea. Due to the depth of the label hierarchy in ChEBI, this family of methods is not applicable to our use case. Preliminary experiments have shown that errors compound for labels further down in the hierarchy, making the Classifier Chain unreliable. Furthermore, the deep learning (DL) models in particular depend on synergistic effects between branches of ChEBI and have proven less effective when pointed to single classes or small subhierarchies of ChEBI.

The second category consists of ensembles based on Label Powerset. Here, the idea is to generate a single-label dataset by treating subsets of the original labels as new labels. Again, this approach does not fit the ChEBI classification task structurally. Given more than 2,000 labels, forming their powerset is not practical, even when discarding a range of combinations based on hierarchy and disjointness constraints. Thirdly, Random Forest of Predictive Clustering Trees [24] is an ensemble that uses Predictive Clustering Trees [25], a variant of decision trees that tries to minimise variation within clusters, as base learners. Decision trees have been applied to ChEBI in previous work [11], but have not been able to match the performance of neural network-based approaches. Moreover, instead of an ensemble technique specialised in one machine learning architecture, we believe that the combination of different methods is crucial for our task.

This leads to the final category proposed by [20], which consists of ensembles that directly combine multi-label base learners. While the other categories use binary or multi-class models that are combined into a multi-label ensemble, here, the base learners themselves are multi-label models. This has several advantages: it allows any model architecture to be used as a base learner, paving the way for heterogeneous ensembles and, in our case, models that are not learned at all. Also, it is easier to

capture inter-label dependencies, which are a common problem for Binary Relevance. Here, the task of capturing dependencies is delegated to the base learners. Tahir et al. [26] have shown that this type of ensemble is well-suited for imbalanced datasets and correlated classes. This is particularly useful in hierarchical multi-label classification. There are two ways to aggregate base learners in this setting: either via weighted majority voting or via a learned meta-model.

2.1.1 Weighted Majority Voting

The simplest form of voting is majority voting (MV), where the classification is based on a majority decision of the base learners. However, especially in heterogeneous ensemble settings, it is reasonable to assume that not all classifiers are equally strong or equally reliable across the target space. To address this issue, several approaches have proposed some form of Weighted Majority Voting (WMV).

Dogan and Birant [27] describe a Weighted Majority Voting Ensemble (WMVE) in which weights are calculated by a reward mechanism. For each correctly classified instance on the validation set, a model's weight is increased by a given amount. This amount depends on how many classifiers have classified this instance correctly (the reward is highest if no other classifier made a correct prediction). In the evaluation, it has been shown that a combination of different machine learning algorithms outperforms individual algorithms for most evaluation datasets. A limitation of this approach is that it has been designed for multi-class problems and only assigns one weight per model. This does not address the fact that model performance can vary heavily depending on the target label. Accounting for this would require an adaptation of WMVE. In addition, the reward mechanism depends on the validation set size and cannot give rewards to one model independently of other models.

Sterniczuk and Charytanowicz [28] have developed an ensemble of convolutional neural networks (CNNs) for the classification of brain tumours. As an aggregation strategy, they use F1-weighted MV. The F1 scores are calculated per model and label and are multiplied by the model predictions. In their evaluation, their F1-weighted majority voting is shown to outperform, among others, MV and WMVE.

The WMV strategy applied in this work is an extension of the voting mechanism proposed in [28]. We adapt the F1-based voting to a multi-label task and add tunable hyperparameters that allow the weights to be calibrated for a given task.

2.1.2 Learned Meta-models

Dinca et al. [29] have developed an ensemble for a multi-class classification problem consisting of different vision transformer architectures. They use the logits output by the base learners (not applying a softmax) and weight them by the validation F1 score for each model and class. Finally, they use the weighted logits as input for a logistic regression model.

You et al. [30] propose GOLabeler, an ensemble of heterogeneous classifiers for predicting Gene Ontology classes. The Gene Ontology [31, 32] is an ontology for annotation of gene products (e.g., proteins). They adapt the learning to rank (LTR) [33] paradigm, which is also the subject of this work. While LTR was originally developed for ranking web pages, they turn it into a multi-label ensemble method by replacing

queries with protein sequences and documents with Gene Ontology classes. A similar application of LTR is DrugE-Rank [34], where six similarity-based methods for identifying drug-target interactions are combined into an ensemble.

Another family of ensemble aggregation techniques is dynamic ensemble selection (DES) [35, 36]. In DES, the most suitable base learner is selected not based on global performance, but on performance on similar samples. For this, the distance to the query sample is measured on a reference sample set, and the performance of each base learner on the closest neighbours is used to estimate its performance on the query sample. META-DES [37] uses a meta-learner to predict the *competence* of each base learner based on the learners' predictions for the query sample as well as their historic performance for neighbouring samples. In [38], the authors hybridise a purely weighting-based and a purely selection-based META-DES approach, creating META-DES.H. In this procedure, a subset of base learners is first selected based on their competence value (assigned by the meta-learner) and a tuned threshold. Then, a WMV is performed with the remaining models, using the competence values as weights. In the evaluation, it is shown that META-DES.H performs better than either the selective or the weighting approach on 30 binary and multi-class classification datasets. They also identify Naive Bayes as the superior meta-classifier compared to multilayer perceptrons, support vector machines and random forests.

In this work, we adapt both LTR and META-DES.H to the ChEBI classification task. They are suitable candidates since they both allow for a fine-grained weighting of predictions. While the adaptation of LTR is straightforward (replacing proteins with molecules and the Gene Ontology with ChEBI), for META-DES.H, we have extended the approach with a distance metric suitable for molecule data (Tanimoto fingerprint similarity). In addition, we have replaced the binarisation of base learner predictions at a fixed threshold (0.5) with a tuned binarisation . Furthermore, since the task discussed in this work is a multi-label problem, we replaced the single competence values per base learner with a label-wise vector. This allows base learners to be weighted differently for different parts of the label hierarchy (similar to WMV).

2.2 Inconsistency Resolution

Most methods for post-hoc inconsistency resolution focus exclusively on *subsumption violations*, i.e., predictions where a superclass is not predicted while a subclass is predicted [39–42]. For this work, we are interested in frameworks that can also model *disjointness violations*, i.e., cases where two positive predictions are not compatible with each other.

Hierarchy and exclusion (HEX) graphs [16], a method based on conditional random fields, are able to identify those violations by interpreting predictions for individual labels as probabilities in order to calculate the most probable combination of predictions that is compatible with the subsumption and disjointness constraints. The authors apply this methodology to ImageNet [43], interpreting all non-overlapping label pairs as exclusion pairs for which they enforce disjointness.

Iterative Local Refinement (ILR) [17] is a fuzzy logic-based approach that aims to align neural network predictions with arbitrary logical formulas. Given a fuzzy t-norm, they use differentiable refinement functions to correct inconsistent predictions with

minimal changes. They give analytical results for three t-norms (Gödel, Łukasiewicz, Product), demonstrating that each t-norm leads to a different refinement behaviour given the same logical formulas.

In this work, we apply both HEX graphs and ILR. One challenge in adapting these methods is the input format: Both methods have been developed for single model predictions that are normalised in the range $[0, 1]$. However, one of the ensemble aggregation methods in this work (LTR) does not produce normalised predictions. For this reason, a normalisation step has been added in which a parametrised sigmoid function is fine-tuned. Furthermore, the HEX graph algorithm depends on predictions being centred around a 0.5 threshold. Since the ensemble methods each tune their own thresholds, this work adds a normalisation step that rescales the ensemble predictions.

For HEX graphs, further adaptations are necessary. Due to the different structure of ChEBI compared to ImageNet, we cannot assume that all classes that do not overlap in the dataset are necessarily disjoint. Instead, this work relies on a set of manually curated, explicit disjointness constraints. As a consequence, the number of legal label combinations becomes intractable. This is alleviated by approximating the original HEX graph resolution with a bounded inference algorithm.

In addition to adapting existing inconsistency resolution methods, this work proposes a 3-step resolution algorithm that is specialised for inconsistency resolution with disjointness and subsumption axioms.

3 Methodology

The architecture of our proposed ensemble model is shown in Figure 2. Roughly, the architecture can be divided into two stages. Stage 0 (shown in yellow) covers the base learners that work directly on the input data. Stage 1 (shown in green) is a meta-model that takes the outputs of base learners as input and turns them into the final prediction. For each input sample x, each base learner m yields

$$\mathbf{p^m}(x) \in ([0, 1] \uplus \{\text{None}\})^C \tag{1}$$

where C is the set of labels covered by the ensemble. For a set X, X^C denotes the function space of all functions $C \to X$. Given a label c, we write $p_c^m(x)$ for $\mathbf{p^m}(x)(c)$. If a base learner reports "None" for a given label, this means that it is either not applicable to this label (e.g., because it was only trained on a subset of labels) or failed to make a prediction for this input (e.g., due to a preprocessing error).

For aggregation methods that require a binarisation of predictions (in this study, all except for LTR), the optimal binarisation threshold θ^m is tuned for every model on the validation set. The threshold that optimises the validation macro F1 score is selected via grid search. This allows assigning each class-wise prediction $p_c^m(x)$ a direction $d_c^m(x)$:

$$d_c^m(x) = \begin{cases} 1 & \text{if} \quad p_c^m(x) > \theta^m \\ 0 & \text{if} \quad p_c^m(x) \leq \theta^m \\ \text{None} & \text{if} \quad p_c^m(x) = \text{None} \end{cases} \tag{2}$$

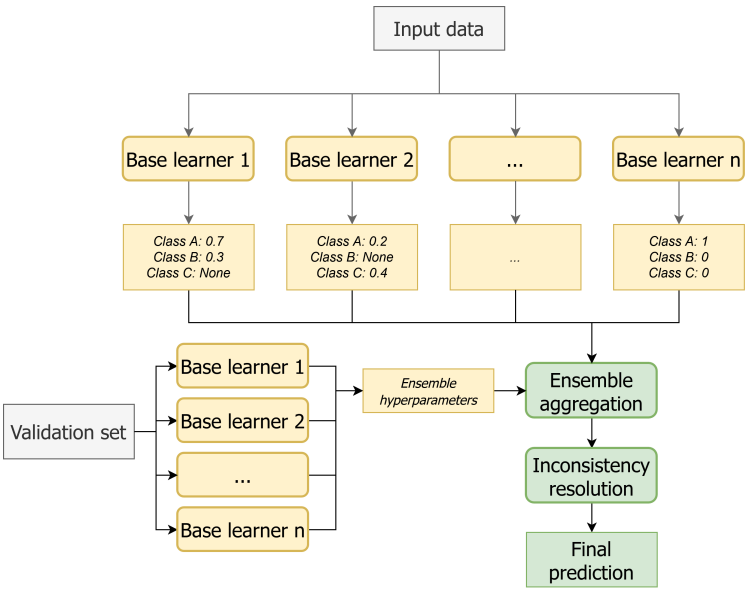


Fig. 2: Architecture of our ensemble. Stages 0 and 1 are highlighted in yellow and green, respectively. Grey is used for inputs to the ensemble. The classes A–C are illustrative and do not refer to actual classes.

This ensures that the direction values of different models are comparable when used as input to the ensemble aggregation. Based on these inputs, the ensemble produces an aggregate prediction vector $\mathbf{p^e}(x) \in [0,1]^C$ (cf. Section 3.1).

In addition, the class-wise predictions produced by the fitted meta-model are verified against the label hierarchy and corrected by an inconsistency resolution algorithm (cf. Section 3.2), producing resolved predictions $\mathbf{p^r}(x) \in [0,1]^C$.

For the binarisation of ensemble predictions, a threshold θ^e is used. For voting methods, this threshold is set to a fixed 0.5 since the predictions are aligned to this value by construction. For the learned meta-models, this threshold is tuned based on the validation macro F1. The ensemble prediction's direction $d_c^e(x)$ is defined via θ^e:

$$d_c^e(x) = \begin{cases} 1 & \text{if} \quad p_c^r(x) > \theta^e \\ 0 & \text{otherwise} \end{cases} \tag{3}$$

Other important factors are the hyperparameters of the meta-model. Depending on the model, different task-specific parameters have to be tuned. This is done via a 5-fold cross-validation on the validation set. For each holdout fold, the meta-model is fitted on the remaining folds and evaluated on the holdout fold. The optimal configuration is then used to fit the meta-model on the complete validation set.

In the following, we first discuss variants for the ensemble aggregation stage (Section 3.1) and subsequently the proposed inconsistency resolution algorithms (Section 3.2). Throughout this section, a running example will be used based on

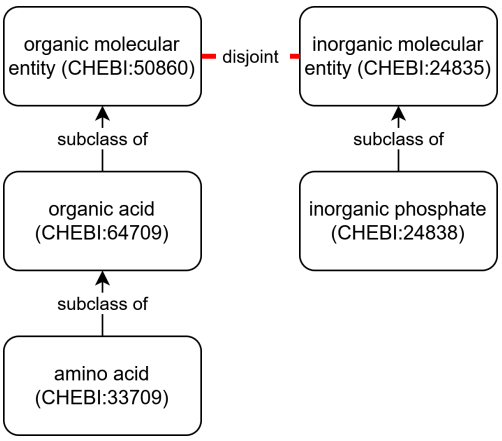


Fig. 3: Example for subsumption and disjointness relations in the ChEBI label hierarchy.

Table 1: Example predictions by fictional base learners m_1, m_2, m_3 for the label hierarchy in Figure 3.

ChEBI class c	$p_c^{m_1}(x)$	$d_c^{m_1}(x)$	$p_c^{m_2}(x)$	$d_c^{m_2}(x)$	$p_c^{m_3}(x)$	$d_c^{m_3}(x)$
organic molecular entity	0.70	1	0.48	1	0	0
organic acid	0.90	1	0.15	0	None	None
amino acid	0.30	0	0.28	1	None	None
inorganic molecular entity	0.40	0	0.15	0	1	1
inorganic phosphate	0.45	0	0.36	1	None	None

an excerpt of the ChEBI hierarchy (cf. Figure 3) with (fictional) base learner predictions by two deep learning models and one rule-based model. As thresholds, $\theta^{m_1} = \theta^{m_3} = 0.5$ and $\theta^{m_2} = 0.2$ are assumed. This means that, for instance, the prediction of m_2 for *inorganic phosphate* (0.36) is counted as positive despite being lower than the negative prediction of m_1 for the same class (0.45).

3.1 Ensemble Aggregation

The aggregation methods used to combine base learners into a single ensemble prediction fall into two categories: Weighted Majority Voting (WMV), which weights base learners according to their performance on a validation set (i.e., data that has not been used during training), and meta-learning, where a meta-model is trained on the outputs of base learners for the validation set.

3.1.1 Weighted Majority Voting

Majority Voting (MV) is a standard technique to aggregate predictions of base learners [44]. In a multi-label classification problem, this means that for each class, the

positive predictions are counted. If the majority of base learners makes a positive prediction, the ensemble also predicts this class.

All voting-based aggregation methods in this work share the same structure: a weighted share of positive votes among a set of base learners. We therefore define a generic *weighted vote* first. Let $\mathbf{m_c^+}(x)$ denote the set of all base learners with $d_c^m(x) = 1$ for class c, $\mathbf{m_c^-}(x)$ the set of base learners with $d_c^m(x) = 0$, and $M_c(x) = \mathbf{m_c^+}(x) \cup \mathbf{m_c^-}(x)$ the set of all base learners that have made a prediction for class c. Given a set of base learners $M \subseteq M_c(x)$ and weights $w^m \geq 0$, the weighted vote for class c is

$$\text{vote}_c(x; M, w) = \begin{cases} \dfrac{\sum_{m \in M \cap \mathbf{m_c^+}(x)} w^m}{\sum_{m \in M} w^m} & \text{if the denominator is } > 0 \\ 0.5 & \text{otherwise.} \end{cases} \tag{4}$$

The fallback value of 0.5 applies if no base learner has made a prediction for class c or if all weights are 0. Majority voting is the weighted vote over all base learners with uniform weights:

$$p_c^{\text{mv}}(x) = \text{vote}_c(x; M_c(x), 1). \tag{5}$$

However, if the base learners have varying performance, this type of voting is not ideal [27]. Two unreliable base learners might outvote a trustworthy one. This can be counteracted by assigning weights that give trustworthy base learners a larger share in the vote compared to less reliable ones, leading to Weighted Majority Voting (WMV) [45]. In this work, we need to balance a wide range of different base learners with different performances and different types of outputs (i.e., continuous outputs for deep learning models and binary outputs for rule-based ones). Therefore, choosing sensible weights is crucial to the ensemble.

WMV (Confidence)

A straightforward approach is to use the predictions themselves as weights. If a base learner makes a prediction close to the decision threshold θ^m, one can interpret this as the model being "not confident" in its prediction. Weighted against another base learner that makes a "confident" prediction close to 0 or 1, it is sensible to stick with the more confident one:

$$\text{conf}_c^m(x) = \begin{cases} \frac{p_c^m(x) - \theta^m}{1 - \theta^m} & \text{if} \quad p_c^m(x) \geq \theta^m \\ \frac{\theta^m - p_c^m(x)}{\theta^m} & \text{if} \quad p_c^m(x) < \theta^m \\ 0 & \text{if} \quad p_c^m(x) = \text{None} \end{cases} \tag{6}$$

measures the confidence of a base learner prediction.

The ensemble prediction thus becomes

$$p_c^{\text{wmv-conf}}(x) = \text{vote}_c\big(x; M_c(x), \text{conf}_c^m(x)\big). \tag{7}$$

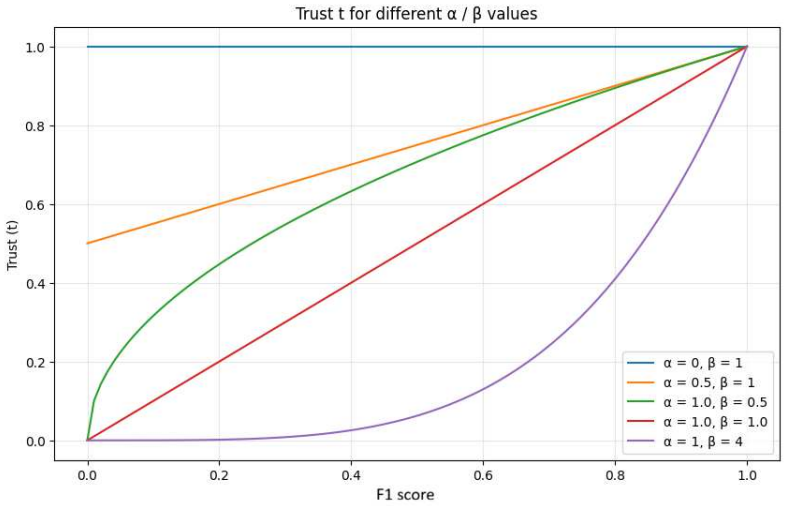


Fig. 4: Trust t in relation to the underlying performance metric (F1-score) for different hyperparameter settings.

WMV (F1-based)

This weighting still lacks information: it cannot estimate whether the model is reliable for a given class or not. Instead of trusting the model output, a more sophisticated weighting incorporates an external validation.

In this work, this type of weight is referred to as *trust*, denoted $\mathbf{t^m} \in [0,1]^C$. It is based on the class-wise F1 score of base learners on a validation set, independently of the current sample [28, 29]. The ensemble multiplies the trust weight with the confidence weight:

$$p_c^{\text{wmv-f1}}(x) = \text{vote}_c\big(x; M_c(x), \text{conf}_c^m(x)\, t_c^m\big). \tag{8}$$

Trust is defined as

$$t_c^m = (\alpha \times \text{F1}_c^m - \alpha + 1)^\beta = (1 - \alpha \times (1 - \text{F1}_c^m))^\beta \tag{9}$$

where α and β are hyperparameters and $\mathbf{F1^m} \in [0,1]^C$ is the model's class-wise F1 score. The F1 score is defined as

$$\text{F1}_c^m = \frac{2TP_c^m}{2TP_c^m + FP_c^m + FN_c^m} \tag{10}$$

where TP_c^m, FP_c^m and FN_c^m are the numbers of true positives, false positives and false negatives across the validation set.

The hyperparameter α determines how "strong" the impact of the F1 score should be. $\alpha = 0$ disables trust completely. A higher β value favours high-performing base learners (i.e., where the F1 score is close to 1) more strongly than a lower β value (cf.

Table 2: Aggregation results for the example predictions from Table 1 using MV aggregation.

ChEBI class c	$p_c^{\text{mv}}(x)$	$d_c^{\text{mv}}(x)$
organic molecular entity	$\frac{2}{3}$	1
organic acid	0.5	0
amino acid	0.5	0
inorganic molecular entity	$\frac{1}{3}$	0
inorganic phosphate	0.5	0

Table 3: Aggregation results for the example predictions from Table 1 using WMV-Conf aggregation.

ChEBI class c	$\text{conf}_c^{m_1}(x)$	$\text{conf}_c^{m_2}(x)$	$\text{conf}_c^{m_3}(x)$	$p_c^{\text{wmv-conf}}(x)$	$d_c^{\text{wmv-conf}}(x)$
organic molecular entity	0.4	0.35	1	$\frac{3}{7}$	0
organic acid	0.8	0.25	0	$\frac{16}{21}$	1
amino acid	0.4	0.1	0	0.2	0
inorganic molecular entity	0.2	0.25	1	$\frac{20}{29}$	1
inorganic phosphate	0.1	0.2	0	$\frac{2}{3}$	1

Figure 4). The optimal values for α and β depend on the task and the base learners used. Preliminary experiments have shown that a larger ensemble size favours higher α and β values, while smaller ensembles work well with lower values of α and β. Both parameters are tuned via cross-validation on the validation set.

Example

One can apply all three voting methods to the example predictions from Table 1. For MV, the procedure is simple: Here, the positive and negative directions get counted per class. For instance, *organic molecular entity* has 2 positive directions (m_1 and m_2) and 1 negative one (m_3). Thus, $p_{ome}^{\text{mv}}(x) = \frac{2}{3}$ is counted as a positive prediction. In the case of ties (e.g., *organic acid*), the prediction counts as negative. Overall, only *organic molecular entity* has a majority of votes and is the only positive prediction by this aggregation method (cf. Table 2).

For confidence-based WMV, we add confidence weights to the vote. These are calculated in relation to the threshold. For instance, *organic acid* receives a fairly confident positive vote from m_1: The prediction of 0.9 is far away from the threshold (0.5), which translates to a confidence of 0.8. m_2 rejects the class, but with less confidence: The 0.15 prediction is close to the threshold (0.2) and thus receives a confidence of only 0.25. Since the third model did not vote on this class, the more confident model gets its way, tipping the balance towards an overall positive prediction ($\frac{16}{21}$). The overall decision are different from MV for 4 out of 5 classes (cf. Table 3): *Organic*

Table 4: Aggregation results for the example predictions from Table 1 using WMV-F1 aggregation. The trust values are example values. Assuming $\alpha = \beta = 1$, they correspond to the validation F1 score of each model.

ChEBI class c	$t_c^{m_1}$	$t_c^{m_2}$	$t_c^{m_3}$	$p_c^{\text{wmv-f1}}(x)$	$d_c^{\text{wmv-f1}}(x)$
organic molecular entity	0.9	0.8	0.96	0.4	0
organic acid	0.75	0.6	-	0.8	1
amino acid	0.9	0.9	-	0.2	0
inorganic molecular entity	0.5	0.4	0.8	0.8	1
inorganic phosphate	0.9	0.3	-	0.4	0

molecular entity is now negative (due to the confident vote of m_3, which always has a confidence of 1 if it votes for a class). Besides *organic acid*, *inorganic molecular entity* and *inorganic phosphate* are now positive as well.

Finally, we include the F1-based weights. For this example, parameters $\alpha = \beta = 1$ and the trust values shown in Table 4 are used. The weight of each model-label pair now consists of confidence and trust. For instance, the prediction of *inorganic phosphate* now calculates as $\frac{0.2 \times 0.3}{0.1 \times 0.9 + 0.2 \times 0.3} = 0.4$. Unlike WMV-Conf, this is a negative prediction. In other words: Despite m_1 not being very confident (0.1) in its rejection of this class, based on prior experience (a validation F1 score of 0.9), the ensemble trusts its prediction. m_2, on the other hand, despite being a bit more confident (0.2), has a bad track record (a validation F1 of 0.3) and therefore loses the vote. Apart from *inorganic phosphate*, the decisions stay unchanged compared to WMV-Conf (cf. Table 4).

3.1.2 Meta-learning

An alternative to WMV are meta-models learned from base learner outputs. In particular, this work incorporates two strategies: Dynamic Ensemble Selection (DES) and Learning To Rank (LTR).

Dynamic Ensemble Selection

This strategy is based on META-DES.H [38], a meta-learning DES approach. In DES, the goal of the meta-model is to classify a model as "competent" or "incompetent" given an input sample. The distinguishing feature of DES is that it not only looks at the input sample itself or at a global metric across the whole validation set (as WMV-F1 does), but specifically at samples in the "region of competence", i.e., samples that are similar to the target sample. The idea is that a base learner which has proven effective for similar samples might perform well on the target sample as well, while in other areas of the feature space, other base learners are more competent. In META-DES.H specifically, the meta-model produces a real-valued score δ which is used to first select base learners that are competent enough (i.e., pass a threshold of 0.5). If no base learner is competent, META-DES.H falls back to including all base learners in the ensemble.

With the selected base learners, the same δ score is used to weight the base learners. In addition, META-DES.H inserts a sample selection phase ahead of the meta-model. This phase excludes all samples which have a high degree of consensus, i.e., samples where the proportion of models agreeing on a prediction surpasses a threshold hC (here, $hC = 0.7$). The goal of sample selection is to exclude trivial predictions from the meta-model, allowing it to focus on the controversial cases.

To apply META-DES.H to the ChEBI classification task, several changes are necessary. Firstly, selecting a region of competence requires a similarity metric. Here, we use Tanimoto similarity over Morgan fingerprints [46, 47]. Secondly, Cruz et al. [38] have used META-DES.H only for binary and multi-class problems. For the multi-label case, instead of having one input vector per base learner and sample, we use individual input vectors for each base learner-sample-label combination, essentially treating every label in the ensemble as a binary classification problem. Thirdly, the meta-learner is label- and model-agnostic: it only sees the performance of a base learner in the region of competence, but it does not distinguish which label and base learner it is judging. Therefore, we add a one-hot feature encoding the base learner identity. This allows the meta-model to treat base learners differently, which we hypothesise to be relevant for a fine-grained weighting of predictions, analogous to the WMV-F1 weights. Furthermore, META-DES.H assumes a binarisation threshold of 0.5 for the base learners. Since our base learners use individual thresholds, we scale the predictions accordingly before applying META-DES.H:

$$p'^{m}_{c}(x) = \begin{cases} 0.5 + \frac{p^m_c(x) - \theta^m}{2(1-\theta^m)} & \text{if} \quad p^m_c(x) \geq \theta^m \\ 0.5 \frac{p^m_c(x)}{\theta^m} & \text{otherwise} \end{cases} \tag{11}$$

Given the competence values $\delta^m_c(x)$ calculated by the meta-model for each base learner m, the selection step retains only the base learners deemed competent for class c (if no competent base learner exists, all base learners are retained).

$$M^{\text{sel}}_c(x) = \{m \in M_c(x) \mid (\delta^m_c(x) > 0.5) \vee \neg\exists m' \in M_c(x) : (\delta^{m'}_c(x) > 0.5)\}. \tag{12}$$

The ensemble prediction is the weighted vote (Equation 4) over the selected base learners, using the competence values as weights:

$$p^{\text{des}}_c(x) = \text{vote}_c\big(x; M^{\text{sel}}_c(x), \delta^m_c(x)\big). \tag{13}$$

Lastly, we add a confidence variant that, in the same vein as WMV-Conf, weights the competence scores $\delta^m_c(x)$ by confidence scores $\text{conf}^m_c(x)$:

$$p^{\text{des-conf}}_c(x) = \text{vote}_c\big(x; M^{\text{sel}}_c(x), \text{conf}^m_c(x)\, \delta^m_c(x)\big). \tag{14}$$

The choice between $p^{\text{des}}_c(x)$ and $p^{\text{des-conf}}_c(x)$ is evaluated as a hyperparameter. Therefore, in the evaluation, we use "DES" to refer to the DES-Conf variant (which has been chosen based on hyperparameter optimisation).

Table 5: Voting-based aggregation methods as instances of the weighted vote (Equation 4). The voting methods differ only in their weights, DES additionally in the set of base learners taking part in the vote.

Method	Base learners M	Weight w^m
MV	$M_c(x)$	1
WMV-Conf	$M_c(x)$	$\text{conf}_c^m(x)$
WMV-F1	$M_c(x)$	$\text{conf}_c^m(x)\, t_c^m$
DES	$M_c^{\text{sel}}(x)$	$\delta_c^m(x)$ *or* $\text{conf}_c^m(x)\, \delta_c^m(x)$

Table 6: Aggregation results for the example predictions from Table 1 using DES aggregation (with confidence). The competence values δ are example values. δ values passing the competence threshold (0.5) and therefore influencing the vote are highlighted in bold.

ChEBI class c	$p'^{m_2}_c(x)$	$\delta_c^{m_1}(x)$	$\delta_c^{m_2}(x)$	$\delta_c^{m_3}(x)$	$p_c^{\text{des-conf}}(x)$	$d_c^{\text{des-conf}}(x)$
organic molecular entity	0.675	**0.9**	**0.6**	0.4	1	1
organic acid	0.375	**0.8**	**0.6**	-	$\frac{64}{79}$	1
amino acid	0.55	**0.7**	0.2	-	0	0
inorganic molecular entity	0.375	**0.7**	**0.6**	0.4	0	0
inorganic phosphate	0.6	**0.8**	0.3	-	0	0

For the final output (that is, after inconsistency resolution has been applied), predictions are binarised via the threshold θ^{des}. Besides the choice between p_c^{des} and $p_c^{\text{des-conf}}$, the parameters $K \in \{1, 7\}$ (size of the region of competence in the feature space), $K_p \in \{5, 9, 15\}$ (profile size in the output space) and $\theta^{\text{des}} \in (0, 1)$ are tuned via cross-validation on the validation set. The grid is based on the default values $K = 7$ and $K_p = 5$ reported by Cruz et al. [37, 38]; $K = 1$ serves as a control in which the region of competence shrinks to the single most similar molecule. As the meta-model architecture, Naive Bayes is used, which Cruz et al. [38] report as best-performing in their evaluation.

Table 5 summarises the voting-based aggregation methods. The three voting methods differ only in their weights, each adding a source of information: WMV-Conf adds the confidence of the individual prediction, WMV-F1 adds the validation performance of the base learner. DES additionally restricts the vote to the base learners selected as competent for the given sample.

Example

As for the WMV variants, we apply DES to the example introduced in Table 1. As a preliminary step, the predictions are normalised to a common threshold of 0.5 (which

only affects m_2, cf. Table 6). Also, instead of the F1-based weights, competence values δ are generated by a model trained on the validation set. The prediction for x takes (apart from the base learner predictions themselves) validation samples into account that are similar to x in the input / output space. For the sake of the example, we assume the values shown in Table 6. Here, m_1 is deemed to be the most reliable model, while m_2 and m_3 get lower scores. m_3 even falls below the competence threshold for both classes, meaning it does not factor into the final prediction. This indicates that, despite scoring high on the classes it predicts (cf. the F1 scores in Table 4), it does not do well in the immediate neighbourhood of x. Instead, DES mostly trusts the predictions of m_1, making positive predictions for *organic molecular entity* and *organic acid* (cf. Table 6).

Learning To Rank

Another meta-learning strategy is Learning To Rank (LTR). The application of LTR to a multi-label classification problem is based on a similar application for the Gene Ontology [30]. Originally, LTR was an information retrieval algorithm for ranking web pages given a query. Here, we replace the query with an input sample (i.e., a molecule structure) and the web pages with ChEBI labels.

LTR is applied only to selected labels. For each learner, the labels with the top-k predictions get chosen. The union of these labels is referred to as the *candidate set* *Cand*. In other words, LTR only handles labels that are in the top-k predictions of at least one base learner. This reduces the complexity of the ranking task and filters out a large number of uncontroversial negatives. The LTR meta-model receives one feature vector per molecule-label pair. Note that, in contrast to META-DES.H and WMV, the meta-model does not act independently for each base learner and is therefore able to capture dependencies between learners. As the meta-model, we use Multiple Additive Regression Trees [48] with the same settings as [30].

Besides the adaptation from the Gene Ontology to ChEBI, the LTR application in this work differs from [30] in several important points: In [30], each feature vector consists of the base learner predictions $\mathbf{p^m}$. In this work, we extend the feature vector with aggregates (including the number of base learners that have produced predictions for this class) and the $\mathbf{F1^m}$ scores (which are also used for the WMV-F1 trust calculation). This allows LTR to compensate for heterogeneous base learners that do not cover the same output space and provides additional information on each model's validation performance.

Furthermore, [30] does not assign a fixed threshold to binarise the meta-model's output. Instead, the F-max score is reported, essentially choosing the optimal threshold on the test set. In this work, the threshold θ^{ltr}, as for the other meta-models, is tuned on the validation set, allowing for a more meaningful test set evaluation. In addition, we apply Platt scaling [49] to the meta-learner output. Platt scaling is a parameterised transformation method based on logistic regression that turns the arbitrary outputs of the meta-model into probabilities that the inconsistency resolution methods can handle. Finally, in [30], non-candidates (i.e., labels that are not in the candidate set *Cand*) are discarded completely. However, for this work, it is necessary to assign

Table 7: Aggregation results for the example predictions from Table 1 using LTR aggregation. The meta-level outputs $r_c(x)$ are example values. Predictions are calculated based on Platt parameters $a = 2$, $b = -1$.

ChEBI class c	$r_c(x)$	$p_c^{\text{ltr}}(x)$	$d_c^{\text{ltr}}(x)$
organic molecular entity	0.6	0.55	1
organic acid	1.4	0.86	1
amino acid	-	0.05	0
inorganic molecular entity	0.7	0.60	1
inorganic phosphate	0.3	0.40	0

prediction values to all labels for inconsistency resolution. Here, the probability that a non-candidate is actually a false negative is used.

Thus, the prediction calculation for LTR works as follows:

$$p_c^{\text{ltr}}(x) = \begin{cases} \sigma(a \times r_c(x) + b) & \text{if} \quad c \in \mathit{Cand} \\ \frac{n_{\text{missed}}+0.5}{n_{\text{non-cand}}+1} & \text{otherwise} \end{cases} \tag{15}$$

where $r_c(x)$ is the raw meta-model output, a and b are the Platt scaling parameters, n_{missed} is the number of false negative non-candidates and $n_{\text{non-cand}}$ is the total number of non-candidates (calculated across the validation set). The parameters $k \in \{30, 50, 70\}$ (candidate set size per base learner), θ^{ltr} and Platt a and b are tuned on the validation set.

Example

Applying the running example to LTR, we first have to identify the candidate set *Cand*. Assuming $k = 2$, the candidate set consists of the two classes with the highest prediction values per model. For instance, for m_1, the highest predictions were 0.9 (*organic acid*) and 0.7 (*organic molecular entity*). Therefore, *organic acid* and *organic molecular entity* are in the candidate set. Besides that, *inorganic phosphate* (due to m_2) and *inorganic molecular entity* (due to m_3) enter the candidate set. Note that *amino acid* is not in the candidate set. This means it will not be predicted by the meta-model but instead receive the default prediction value based on the number of false negatives among non-candidates (for the sake of the example, we assume this ratio to be 0.05).

The remaining classes get predicted by the meta-model. As input, it receives a feature vector consisting of predictions, aggregates and F1 scores. For example, *organic acid* uses the following feature vector: (0.9, 0.15, None, 2, 0.9, 0.525, 0.53, 0.75, 0.6, 0, 0.12, 646), which corresponds to prediction values per model, number of valid predictions, maximum, mean, standard deviation, F1 score per model (cf. Table 4), prevalence and number of positives (out of 5,313 validation samples, 646 are organic acids). Table 7 shows fictional meta-model outputs. Assuming Platt parameters $a = 2$,

$b = -1$ and a threshold $\theta^{\text{ltr}} = 0.5$, the model predicts *organic molecular entity*, *organic acid* and *inorganic molecular entity*.

3.2 Inconsistency Resolution

So far, ensemble predictions $\mathbf{p}^{\mathbf{e}}(x)$ are made independently for each class. This means that an ensemble can, for instance, confidently predict that a molecule is an *organic molecular entity*, but at the same time claim that it is an *inorganic molecular entity* (as LTR does in the example above). Such predictions are confusing and not helpful for users.

However, with the ChEBI ontology, we have an extensive set of formalised relations at hand that can be exploited to resolve such inconsistencies automatically. Specifically, two types of inconsistencies are taken into account (cf. Figure 3): (1) the ensemble predicts that a molecule belongs to class A, but not B, in spite of the fact that A is a *subclass* of B (subsumption inconsistency); (2) the ensemble predicts that a molecule belongs both to class A and class B, in spite of the fact that A and B are *disjoint* classes (disjointness inconsistency). Unless stated otherwise, we always use the transitive closure of the subsumption relation, and we propagate disjointness to subclasses: if A and B are disjoint, then every subclass of A is disjoint with every subclass of B. This means that, for instance, given that *organic molecular entity* and *inorganic molecular entity* are disjoint, we infer that *amino acid* and *inorganic phosphate* are disjoint as well.

We apply three algorithms to this problem: Hierarchy and Exclusion (HEX) graphs [16], Iterative Local Refinement (ILR, [17]) and a new algorithm called *3-step inconsistency resolution*.

3.2.1 HEX Graphs

In a HEX Graph $G = (V, E_h, E_e)$, each node $v \in V$ models a label, each directed edge $(v_i, v_j) \in E_h$ a superclass-subclass relation between i and j and each undirected edge $(v_i, v_j) \in E_e$ a disjointness relation between i and j. For such a HEX graph, an assignment of labels $\mathbf{y}^{\mathbf{e}}(x) \in \{0, 1\}^n$ can be defined. $\mathbf{y}^{\mathbf{e}}(x)$ is consistent if it never assigns 0 and 1 to a superclass-subclass pair and never assigns 1 to two disjoint classes. Thus,

$$Y^e = \{\mathbf{y}^{\mathbf{e}} | \forall (v_i, v_j) \in E_h : (y_i^e, y_j^e) \neq (0, 1) \wedge \forall (v_i, v_j) \in E_e : (y_i^e, y_j^e) \neq (1, 1)\} \quad (16)$$

denotes the set of consistent assignments.

Each assignment gets a probability that measures how close the assignment is to the actual ensemble output. For this, we first convert the raw ensemble predictions $\mathbf{p}^{\mathbf{e}}(x)$ to normalised predictions ${p'}_c^e(x)$ using Equation 11 (adapted for the ensemble e instead of base learner m). This means that the decision threshold for the normalised predictions becomes $\theta^{\text{hex}} = 0.5$, which supersedes the ensemble threshold θ^e. The normalisation is necessary because the HEX graph algorithm guarantees consistency only at this threshold: For disjointness relations, the sum of both probabilities cannot exceed 1.

For the HEX graph algorithm, the predictions are first cast to logits $\mathbf{l^e}(x) \in \mathbb{R}^C$:

$$l_c^e(x) = \log {p'}_c^e(x) - \log(1 - {p'}_c^e(x)) \tag{17}$$

The logit is the inverse of the sigmoid function, so that, in the absence of any constraints, the marginals $p_c^{\text{hex}}(x)$ computed below coincide with the inputs ${p'}_c^e(x)$; any deviation is thus caused by the constraints alone. Since the logit is infinite for ${p'}_c^e(x) \in \{0, 1\}$, which occurs in particular for the binary outputs of the rule-based base learners, the normalised predictions are clipped to the interval $[\varepsilon, 1 - \varepsilon]$ with $\varepsilon = 10^{-6}$ beforehand. Based on the logits, we can calculate a probability for every consistent assignment $\mathbf{y^e} \in Y^e$ by applying a softmax:

$$\Pr(\mathbf{y^e}|x) = \frac{w(\mathbf{y^e})}{\sum_{\hat{y}^e \in Y^e} w(\hat{y}^e)} \tag{18}$$

where $w(\mathbf{y^e})$ denotes the weight of an assignment, defined as the product of odds for every positive label in the assignment:

$$w(\mathbf{y^e}) = \prod_{c \text{ where } y_c^e = 1} e^{l_c^e(x)} \tag{19}$$

These probabilities are then marginalised to score the individual labels:

$$p_c^{\text{hex}}(x) = \frac{N_c}{N_c + M_c} \tag{20}$$

where

$$N_c = \sum_{\mathbf{y^e} \in Y^e \text{ where } y_c^e = 1} w(\mathbf{y^e}), \quad M_c = \sum_{\mathbf{y^e} \in Y^e \text{ where } y_c^e = 0} w(\mathbf{y^e}) \tag{21}$$

denote the sum of weights for assignments with and without label c.

For the use case discussed in this work, one cannot apply this method as-is. The underlying reason is the structure of the label space: While Deng et al. [16] apply the method only to multi-class problems (which means that they can assume disjointness between all leaf classes), this work targets a multi-label problem with sparse disjointness relations. This means that there are more consistent assignments in Y^e over which the algorithm would need to iterate. Even when using the junction tree optimisation suggested by [16], for ChEBI, there are more than 2^{30} consistent assignments, making an exact calculation infeasible. Instead, this work proposes a bounded inference algorithm that approximates $p_c^{\text{hex}}(x)$. This algorithm allows identifying the direction of the prediction with certainty (i.e., either the upper bound is below the threshold or the lower bound is above the threshold) for more than 99% of labels within reasonable effort (2,000 iterations). The bounded inference algorithm is discussed in detail in Appendix A. Here, it is explained by means of an example.

Table 8: Consistent assignments, their weights and probabilities (given predictions from Table 4). The label order is *organic molecular entity*, *organic acid*, *amino acid*, *inorganic molecular entity*, *inorganic phosphate*.

$\mathbf{y^e}$	$w(\mathbf{y^e})$	$\Pr(\mathbf{y^e}\|x)$
$[0,0,0,0,0]$	1	$\frac{3}{35} = 0.086$
$[0,0,0,1,0]$	4	$\frac{12}{35} = 0.343$
$[0,0,0,1,1]$	$\frac{8}{3}$	$\frac{8}{35} = 0.229$
$[1,0,0,0,0]$	$\frac{2}{3}$	$\frac{2}{35} = 0.057$
$[1,1,0,0,0]$	$\frac{8}{3}$	$\frac{8}{35} = 0.229$
$[1,1,1,0,0]$	$\frac{2}{3}$	$\frac{2}{35} = 0.057$

Example

We continue the running example with ensemble predictions from WMV-F1 aggregation (cf. Table 4). The ensemble predictions are 0.8 for *organic acid* and *inorganic molecular entity*, 0.4 for *organic molecular entity* and *inorganic phosphate* and 0.2 for *amino acid*. Note that these predictions are not consistent with the ChEBI axiomatisation (cf. Figure 3). *organic acid* and *inorganic molecular entity* violate the disjointness axiom (a molecule cannot be organic and inorganic at the same time); *organic acid* and *organic molecular entity* violate a subsumption axiom (any organic acid is an organic molecular entity). Thus, we need to apply inconsistency resolution.

The HEX graph algorithm starts by calculating the logits using Equation 17. For this example, it is possible to enumerate all valid assignments Y^e directly, which is necessary to calculate $\mathbf{p}^{\text{hex}}(x)$. Table 8 lists all assignments, their weights and their probabilities. $\mathbf{p}^{\text{hex}}(x)$ is the marginalisation of these probabilities. Only *inorganic molecular entity* surpasses the threshold, and *organic acid* has been corrected from a positive to a negative prediction.

While we can calculate $\mathbf{p}^{\text{hex}}(x)$ directly in this case, for demonstration purposes, we also apply the bounded approximation method (cf. Appendix A). The iteration steps are shown in Figure 5. The central idea is, instead of looking at each assignment, to start with *partial assignments* for which upper and lower bounds of the assignment weight can be calculated. The frontier $\mathcal{A}$ is initialised with the default assignment (unknown for all labels). In the first iteration, the label k with the highest $\text{softplus}(l_k^e(x))$ is chosen. Here, *organic acid* and *inorganic molecular entity* both have the maximum value, so we choose *organic acid* at random, creating two new partial assignments, a_1 and a_2. Note that a_1 already fixes four labels: *organic acid* is set as positive by construction; the remaining labels are inferred directly. a_2 has more uncertainty and is therefore split at the *inorganic molecular entity* node in iteration 2, creating a_3 and a_4. At this point, some labels are already certified: for instance,

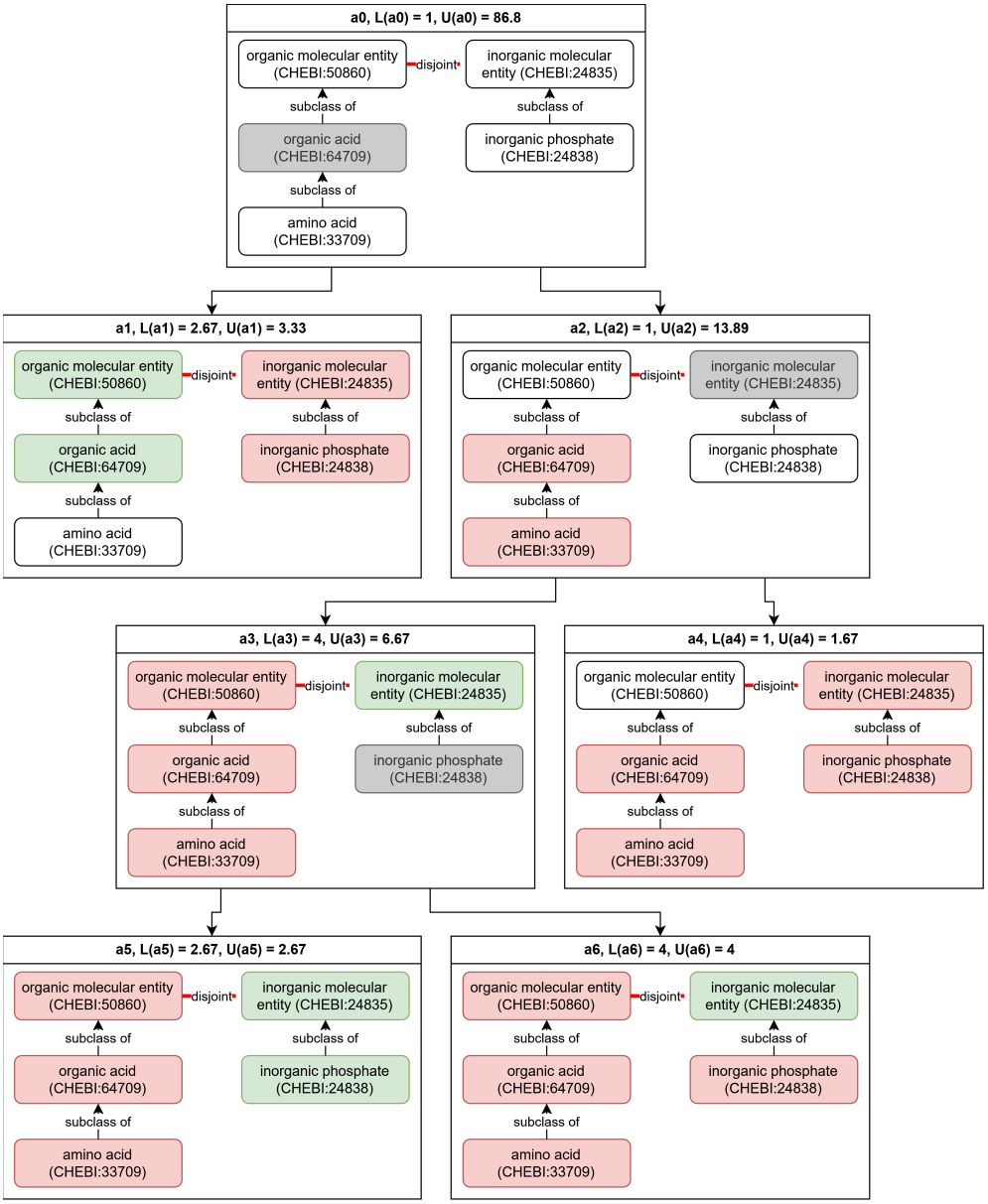


Fig. 5: HEX graph bounded inference (cf. Algorithm 2 of Appendix A) applied to the running example. Each box represents a partial assignment. Labels in the P set of the assignment are shown in green, labels in the Q set are shown in red. Each iteration splits an assignment into two new assignments along the label k^* (highlighted in grey). After 3 iterations, the algorithm has converged.

Table 9: Example calculation for HEX graph inconsistency resolution. The bounds $\pi^{\mathbf{lo}}$ and $\pi^{\mathbf{hi}}$ have converged in 3 iterations (cf. Figure 5).

ChEBI class c	$p_c^e(x)$	$l_c^e(x)$	$p_c^{\text{hex}}(x)$	π_c^{lo}	π_c^{hi}
organic molecular entity	0.4	−0.41	$\frac{12}{35}$	$\frac{10}{35}$	$\frac{15}{38}$
organic acid	0.8	1.39	$\frac{10}{35}$	$\frac{8}{33}$	$\frac{10}{33}$
amino acid	0.2	−1.39	$\frac{2}{35}$	$\frac{2}{37}$	$\frac{10}{41}$
inorganic molecular entity	0.8	1.39	$\frac{20}{35}$	$\frac{20}{35}$	$\frac{20}{31}$
inorganic phosphate	0.4	−0.41	$\frac{8}{35}$	$\frac{8}{35}$	$\frac{8}{31}$

amino acid is set as negative in a_3 and a_4 and undecided in a_1. Thus, the highest weight it may still get is $U(a_1) = \frac{10}{3}$. We also know that the weights it cannot get sum to $L(a_1) + L(a_3) + L(a_4) = \frac{23}{3}$.This sets the upper bound at $\pi_{aa}^{hi} = \frac{10}{33}$, well below the threshold. For all labels to be certified, a final iteration is needed, separating a_3 into a_5 and a_6. These correspond to single assignments, which is why their lower and upper bounds match the weight of that assignment exactly.

Table 9 shows the final lower and upper bounds, which are consistent with $\mathbf{p}^{\text{hex}}(x)$. With that, only *inorganic molecular entity* receives a positive prediction and *organic acid* has been corrected from positive to negative.

3.2.2 Iterative Local Refinement

ILR [17] is a generic algorithm to align machine learning predictions with arbitrary logical formulas. The underlying idea is to use a fuzzy logic interpretation of predictions and constraints to calculate the fuzzy truth value of a constraint. This is compared to the desired truth value and the predictions are adjusted via a backpropagation algorithm. The backpropagation uses *minimal refinement functions* ρ^*, i.e., functions that apply a minimal change to the input predictions such that the formula output matches the desired value. Daniele et al. [17] provide analytical results for several fuzzy t-norms and perform a practical evaluation. Based on their results, in this work, we focus on ILR with either the *Gödel* or the *Łukasiewicz* t-norm. The Gödel t-norm is the minimum function $T_G(a, b) = \min\{a, b\}$. The *Łukasiewicz* t-norm is defined as $T_L(a, b) = \max\{0, a + b - 1\}$.

For the scheduling parameter α, 1 is used. We interpret subsumption relations where A is a subclass of B as fuzzy implications with target value $\hat{t} = 1$. Disjointness relations between labels A and B are interpreted as fuzzy conjunctions with target value $\hat{t} = 0$. The full refinement functions are shown in Appendix B. ILR iteratively applies these refinement functions to all subsumption / disjointness pairs until convergence. In each iteration, the refinement function with the largest absolute change is applied to each label.

Table 10: Example of prediction changes over two ILR iterations (Gödel / Łukasiewicz).

ChEBI class c	$p_c^e(x)$	$p_c^{\text{ILR-G1}}$	$p_c^{\text{ILR-G2}}$	$p_c^{\text{ILR-L1}}$	$p_c^{\text{ILR-L2}}$
organic molecular entity	0.4	0.8	0	0.6	**0.55**
organic acid	0.8	0	0	0.5	0.5
amino acid	0.2	0	0	0.2	0.2
inorganic molecular entity	0.8	0.8	**0.8**	0.5	0.45
inorganic phosphate	0.4	0	0	0.3	0.3

Example

For the running example, we can apply ILR with the Gödel / Łukasiewicz t-norm (cf. Table 10). In the first iteration step, *organic molecular entity* has two relations that would change its prediction by 0.4, but in opposite directions: the subsumption relation with *organic acid* would push it upwards and the disjointness with *inorganic molecular entity* would push it downwards. Thus, the decision is arbitrary (for the example, the first relation is chosen). The other changes are determined by disjointness relations which send all classes to 0 (except for *inorganic molecular entity*, which arbitrarily wins the disjointness relation with *organic acid*). In the second iteration, the disjointness between *organic* and *inorganic molecular entity* is decided (again arbitrarily) in favour of *inorganic molecular entity*, leaving it as the final prediction. Note that several major decisions boil down to the arbitrary label order in this example. While this exact case is unlikely in practice (the ensemble predictions are float values), it still illustrates the drastic decision-making procedure of ILR-Gödel: minor differences in predictions can have a major impact on the results.

With Łukasiewicz, the changes between iterations tend to be gentler: the first iteration bumps *organic molecular entity* up to 0.6 (based on the subsumption with *organic acid*) and reduces all others (except for *amino acid*) slightly based on the disjointness relations. In the second iteration, the excess of *organic* and *inorganic molecular entity* over 1 is split evenly between the two, resulting in 0.55 and 0.45. The lead of *organic molecular entity* is not arbitrary: it stems from the subsumption with *organic acid*, which lifted it in the first iteration, whereas *inorganic molecular entity* had no such support. Note that under the Łukasiewicz t-norm, a disjointness relation is satisfied as soon as $t_A + t_B \leq 1$. After the second iteration, all constraints are fulfilled in this sense, and ILR terminates without declaring either class the loser. This decision is only made by the binarisation: since $t_A + t_B \leq 1$, at most one of two disjoint classes can exceed the threshold of 0.5. Here, *organic molecular entity* is the final prediction, while *inorganic molecular entity* (0.45) and *organic acid* (exactly 0.5) are predicted negative. This result is different from ILR-Gödel. Instead of a "one loses, the other stays" strategy, with ILR-Łukasiewicz, both disjoint classes are corrected downwards, leading to a more balanced result.

Algorithm 1 Inconsistency resolution

Input: $\mathbf{p^e}(x) \in \mathbb{R}^C$
Output: $\mathbf{p^{r3}}(x) \in \mathbb{R}^C$
for all $c \in C$ **do** ▷ Step 1: Resolve subsumption violations
    $S^{sup} \leftarrow \{j | c \sqsubseteq j \wedge p_j^e(x) \leq \theta^e\}$
    $S^{sub} \leftarrow \{j | j \sqsubseteq c \wedge p_j^e(x) > \theta^e\}$
    $j_c^* \leftarrow \underset{j \in (S^{sup} \cup S^{sub})}{\arg\max} \text{conf}_j^e(x)$ ▷ cf. Equation 6
    $p_c^{r1}(x) \leftarrow p_{j_c^*}^e(x)$
end for
$\mathbf{p^{r2}}(x) \leftarrow \mathbf{p^{r1}}(x)$
for all $(a, b) \in \{(a, b) | (a, b) \in C^2 \wedge a \cap b = \bot\}$ **do** ▷ Step 2: Resolve disjointness violations
    if $p_a^{r1}(x) < p_b^{r1}(x)$ **then**
        $p_a^{r2}(x) \leftarrow \min(p_a^{r2}(x), \theta^e)$
    end if
    if $p_b^{r1}(x) \leq p_a^{r1}(x)$ **then**
        $p_b^{r2}(x) \leftarrow \min(p_b^{r2}(x), \theta^e)$
    end if
end for
for all $c \in C$ **do** ▷ Step 3: Propagate to subclasses
    $S^{sup} \leftarrow \{j | c \sqsubseteq j\}$
    $p_c^{r3}(x) \leftarrow \min_{j \in S^{sup}} p_j^{r2}$
end for

3.2.3 3-step Inconsistency Resolution

While HEX graphs and ILR represent generic approaches to resolving subsumption and disjointness inconsistencies (or, in the case of ILR, arbitrary inconsistencies), we additionally propose a more specialised methodology exploiting the ontology structure.

Inconsistencies are resolved in a three-step approach, outlined in Algorithm 1. Step 1 resolves subsumption inconsistencies by always trusting the most confident prediction (in relation to the threshold θ^e) out of the positive sub- and negative superclasses of each class (including the class itself). After step 1, it is guaranteed that $p_A^{r1}(x) \leq p_B^{r1}(x)$ for all label pairs $A \sqsubseteq B$: If $p_A^e(x) > p_B^e(x)$ and both are below the decision threshold, the prediction for B will have a higher confidence value and $p_A^{r1}(x)$ gets set to $p_B^e(x)$. If both are above the decision threshold, A has the higher confidence and $p_B^{r1}(x)$ will get lifted. If both are on opposite sides of the threshold, the less confident one gets overwritten.

In step 2, disjointness inconsistencies are resolved. For each pair of disjoint classes A and B, the class with the lower prediction gets pushed below the threshold (if necessary). For efficiency, only directly asserted disjointness relations are used, not those propagated to subclasses. Thus, for instance, a subclass C of A will not get compared to B, even though it can be inferred from the relation between A and B that it is also disjoint with B. This can lead to new subsumption inconsistencies. Step

Table 11: Example of prediction changes with the 3-step inconsistency resolution.

ChEBI class c	$p_c^e(x)$	$p_c^{r1}(x)$	$p_c^{r2}(x)$	$p_c^{r3}(x)$
organic molecular entity	0.4	0.8	0.8	**0.8**
organic acid	0.8	0.8	0.8	**0.8**
amino acid	0.2	0.2	0.2	0.2
inorganic molecular entity	0.8	0.8	0.5	0.5
inorganic phosphate	0.4	0.4	0.4	0.4

3 propagates the changed predictions to the subclasses, capping their prediction score at the minimum of their superclasses.

Proposition 1. *After executing Algorithm 1, the resulting ensemble prediction* $\boldsymbol{p}^{\mathbf{r3}}(x)$ *is consistent with the ontology.*

A proof sketch for Proposition 1 can be found in Appendix C.

Example

We apply the 3-step inconsistency resolution to the same example used for the other resolution algorithms (cf. Table 11). Step 1 identifies the subsumption violation between *organic molecular entity* and *organic acid*. Since 0.8 is more confident than 0.4 (assuming a threshold 0.5), *organic molecular entity* gets set to 0.8. In the second step, only explicit disjointness relations are taken into consideration, meaning that *organic acid* and *inorganic molecular entity* are not resolved directly. Instead, step 2 only looks at *organic* and *inorganic molecular entity*, deciding in favour of *organic molecular entity* based on the arbitrary label order. Thus, *inorganic molecular entity* gets lowered to 0.5. The final step would correct any remaining subsumption violations, which are not present in this example. This way, the final prediction is *organic molecular entity* and *organic acid*.

Overall, this example has shown different inconsistency resolution results given the same input values. Although this is a deliberately chosen edge case, it illustrates that the resolution algorithm can have a real impact on the ensemble predictions. In Section 5.3, we discuss how the algorithms affect the ensemble in practice.

4 An Ensemble for ChEBI

In the evaluation, the ensemble architecture described in Section 3 will be applied to the ChEBI ontology extension task [11]. The goal of this task is to assign ChEBI classes to molecules based on their structure. For instance, given the molecular structure *CCO* (ethanol), an ideal model would return all ChEBI classes to which this structure may be assigned (e.g., *alkyl alcohol* and *primary alcohol*). This section describes the datasets used for the learning and evaluation steps (Section 4.1) as well as the base learner models (Sections 4.2 and 4.3).

Table 12: Overview of ChEBI datasets

Dataset	Threshold	#Samples	#Labels
$\mathbf{ChEBI}^{\mathbf{2,117}}_{\mathbf{full}}$	25	192,336	2,117
$\mathbf{ChEBI}^{\mathbf{1,766}}_{\mathbf{3-STAR}}$	25	52,901	1,766
$\mathbf{ChEBI}^{\mathbf{2,234}}_{\mathbf{full}}$	-	192,336	2,234
$\mathbf{ChEBI}^{\mathbf{2,234}}_{\mathbf{3-STAR}}$	-	52,901	2,234

4.1 Datasets

Our ensemble includes two types of base learners: deep learning and symbolic ones. The deep learning models have been trained on molecule data from ChEBI, following the approach proposed in [12]. All classes that are annotated with a molecular structure are taken as samples. Classes for which at least a threshold number of subclasses are samples are used as labels. Here, we use a threshold of 25. This results in the $\mathbf{ChEBI}^{\mathbf{2,117}}_{\mathbf{full}}$ dataset with 2,117 labels (cf. Table 12). Note that this already extends the datasets used in previous work ([12] used a threshold of 50 to collect 1,332 labels, [50] collected 1,528 with the same threshold, but a newer ChEBI version). Additionally, in this work, we propose a dataset $\mathbf{ChEBI}^{\mathbf{1,766}}_{\mathbf{3-STAR}}$ that only takes samples from the 3-STAR fragment of ChEBI into account. The 3-STAR fragment contains only manually verified structures. Therefore, it trades higher data quality for lower availability of training data and lower class coverage. The resulting datasets are available in the Supplementary Material.

For each deep learning model, we train two versions: a 3-STAR version trained on $\mathbf{ChEBI}^{\mathbf{1,766}}_{\mathbf{3-STAR}}$ and a full-ChEBI version trained on $\mathbf{ChEBI}^{\mathbf{2,117}}_{\mathbf{full}}$. For both datasets, we apply the same 80/10/10 train/validation/test split (measured on $\mathbf{ChEBI}^{\mathbf{2,117}}_{\mathbf{full}}$) to avoid train-test leakage when evaluating both on the same test set.

Since symbolic base learners also cover ChEBI classes outside the $\mathbf{ChEBI}^{\mathbf{2,117}}_{\mathbf{full}}$ dataset, for the evaluation, we extend both datasets to the set of labels predicted by at least one base learner ($\mathbf{ChEBI}^{\mathbf{2,234}}_{\mathbf{full}}$ and $\mathbf{ChEBI}^{\mathbf{2,234}}_{\mathbf{3-STAR}}$).

Inconsistency resolution is based on subsumption and disjointness relations between labels. For subsumption relations, we use the *is-a* property of the ChEBI ontology. For disjointness relations, we use the *disjoint with* property. Axioms for this property are provided as a separate module [51]. In this work, an extended version of this module is used.[2]

4.2 Deep Learning Models

The Chebifier 2 ensemble incorporates four deep learning architectures as base learners: Residual-Gated Graph Convolutional Networks (RGGCNs), Graph Attention Networks (GATs) and augmented variants of both. All are variants of graph neural networks (GNNs), which have proven effective for cheminformatics tasks in previous

[2] The additional relations are available at [52].

Table 13: Properties used as node initialisation / edge features for graph neural networks. RDKit2DNormalized is a set of 200 properties calculated by RDKit. These properties are used as initialisation for the global node (in augmented graphs) or concatenated with the graph representation after pooling (in non-augmented graphs).

Level	Properties (all graph models)	Properties (only augmented)
Atom	Aromaticity, Charge, Hybridisation, #Hydrogen neighbours, Atom type, #Bonds	Node level (atom / functional group / global)
Bond	Aromaticity, In ring, Bond type	Bond level (atom to functional group / functional group to global)
Functional groups		Functional group, Is alkyl, Is hydrogen bond acceptor, Is hydrogen bond donor
Global	RDKit2DNormalized	

work [53, 54]. The standard variants of RGGCNs and GATs utilise an atom-based encoding of molecular structures: Each atom corresponds to one node in the input representation, each covalent bond to an edge. A list of chemical features used to encode each atom and bond can be found in Table 13. In addition, variants with an augmented representation are trained, combining an atom representation with nodes for functional groups and a global node.

Training has been conducted with the ChEB-AI Graph library [55].

Residual-Gated Graph Convolutional Networks

The first architecture is Residual-Gated Graph Convolutional Networks (RGGCNs) [56]. RGGCNs extend the vanilla graph convolutional network architecture [57] by an edge-gating mechanism and residual connections.

Graph Attention Network

The second set of graph neural networks consists of graph attention networks (GATs) [58]. We use the GATv2 architecture [59], which is more expressive than the vanilla GAT. Also, it has been shown to outperform vanilla GAT on the ChEBI classification task [50]. Unlike RGGCNs (and graph convolutional networks in general), GATs use an attention mechanism for weighting the impact of neighbours.

Augmented GNNs

For both RGGCNs and GATs, we use an augmented variant developed in [50]. This variant introduces a three-layer architecture consisting of an atom layer, which corresponds to the non-augmented graphs, a functional group layer, covering 101 common functional groups and 62 ring structures, as well as a global node. Edges connect atoms and the functional groups they are part of as well as functional groups and the global node. The properties used for each layer can be found in Table 13. Figure 6 shows an example molecule representation in the augmented graph neural network architecture.

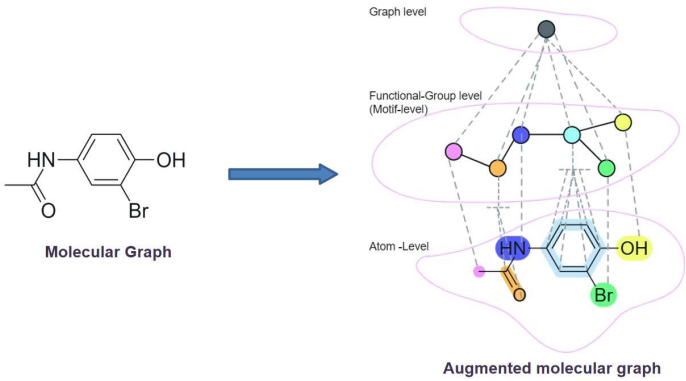


Fig. 6: Pictorial representation of an augmented graph used by GNNs. Adapted from [50].

4.3 Symbolic Models

Apart from the deep learning models, we also include three rule-based models. Chemical Classifier Programs (CCPs) are Python programs generated by large language models (LLMs). In ChemLog, hand-crafted rules in monadic second-order logic (MSOL) are used. Lopster is an answer set programming (ASP) tool relying on hand-crafted rules.

Chemical Classifier Programs

Based on the natural language definitions of ChEBI, LLMs are used to generate CCPs, customised Python scripts that use the RDKit library to classify chemicals [10]. CCPs are generated in an iterative process, where molecules sampled from ChEBI are used to prompt the LLM to refine its original classification program. Alongside the predictions, natural-language explanations for each possible classification result are returned. The implementation is available on GitHub [10]. Programs for the same classes have been generated by different LLMs and with different settings, leading to a total of nine programs for each class. Here, we use the best-performing program for each class (referred to as the ensemble program ontology in the original paper). The CCPs integrated in our Chebifier 2 ensemble cover 729 chemical classes (including classes which are predicted indirectly, i.e., superclasses of classes for which a CCP exists).

ChemLog

ChemLog [60, 61] is a rule-based classification tool described in [7]. It is based on an MSOL formalisation of molecules. Logical definitions are used to generate (and validate) code that classifies molecules automatically. ChemLog covers 18 peptide-related classes discussed in [8]. In this work, we extend ChemLog with 155 classes that are defined by either the presence of a chemical element (e.g., *nitrogen molecular entity*, CHEBI:51143) or by a bond between a carbon atom and a given chemical element (e.g., *organoiron compound*, CHEBI:51185). Two exceptions are *organophosphorus*

compound (CHEBI:25710), which, in addition to compounds with direct carbon-phosphorus bonds, also includes esters and thioesters, and *hydrogen molecular entity* (CHEBI:33608), which requires a different formalisation since hydrogen atoms and other atoms are represented in different ways in ChemLog. These classes have therefore been excluded from the ChemLog formalisation, but are covered by Lopster. Out of the 220 classes predicted by ChemLog in total (including superclasses of directly predicted classes), 109 are not predicted by any of the other models.

Logic Programming for Structured Entities Reasoner (Lopster)

Lopster [9] is a reasoning system for the classification of chemical compounds. In [9], 51 chemical classes are formalised. This includes cyclicity-related classes that cannot be represented in the Web Ontology Language (OWL, [62]), a format based on description logics that is commonly used for ontological reasoning. The original Java implementation of Lopster [63] was based on the disjunctive Datalog implementation DLV [64]. Due to issues with outdated dependencies, instead of using the original implementation, we integrated the rule base of Lopster into ChemLog. As the reasoning component, clingo [65] is used.

The rules have been mapped to 36 ChEBI classes. Since no evaluation of Lopster against ChEBI had been conducted previously, we performed a full classification of ChEBI according to the Lopster rules and compared the results to both the existing ChEBI classification and ChemLog. Errors that were identified in this evaluation have subsequently been fixed. In total, Lopster can make predictions for 62 classes (consisting of directly classified ChEBI classes and their superclasses).

5 Evaluation and Discussion

In this section, the Chebifier 2 ensemble is evaluated. First, we compare the base learners on their own, highlighting important structural differences between deep learning and symbolic models (Section 5.2). Subsequently, we move on to the aggregation methods and inconsistency resolution algorithms (Section 5.3). For the best configurations, DES + ILR-Łukasiewicz and WMV-F1 + 3-step resolution, we take a closer look at the results, identifying strengths and weaknesses of the ensemble (Section 5.4) and attributing the performance gains to the individual base learners (Section 5.5).

5.1 Setup

All deep learning (DL) models use four hidden layers and have been trained on one of the datasets described in Section 4.1. All models have been trained with Asymmetric Loss [66] ($\gamma^+ = \gamma^- = 1$, $m = 0.15$) and a learning rate of 2×10^{-3}. The choice of loss function and parameters used are based on the results of [67]. For each training run of 100 epochs, the model checkpoint at the epoch with the highest validation macro F1 has been selected.

The main evaluation metrics are the F1 score (cf. Eq. 10) and the consistency rate. The F1 score is the harmonic mean of precision ("Out of all positive predictions, how many are correct?") and recall ("Out of all positive labels, how many were predicted?"). There are two possible aggregations of the F1 score across multiple classes:

Table 14: Base learner performance on the test split of $\mathbf{ChEBI^{2,234}_{3-STAR}}$. #Labels is the total number of labels the model can make predictions for. "missing excluded" excludes (molecule, label) pairs for which a model did not produce a prediction, "missing = negative" counts them as 0. F1 scores and consistency are given in percent.

		missing excluded		missing = negative		Consis-
Base learner	#Labels	micro F1	macro F1	micro F1	macro F1	tency
GAT-aug (3-STAR)	1,766	**91.85**	74.82	**91.66**	59.15	68.89
GAT (3-STAR)	1,766	90.98	73.76	90.79	58.31	60.17
RGGCN-aug (3-STAR)	1,766	91.44	73.71	91.25	58.27	67.54
RGGCN (3-STAR)	1,766	90.89	73.26	90.71	57.91	60.16
GAT-aug	2,117	89.07	68.56	89.06	**64.96**	62.71
GAT	2,117	86.54	62.73	86.52	59.45	47.50
RGGCN-aug	2,117	88.98	66.14	88.97	62.68	63.94
RGGCN	2,117	86.53	62.11	86.52	58.86	49.92
Lopster	62	82.87	**74.94**	51.00	1.80	**100.00**
CCPs	729	73.23	59.06	59.95	15.49	79.86
ChemLog	220	80.01	48.32	51.91	4.23	**100.00**

the **micro F1** sums up all true positives, false positives and false negatives across all classes and then calculates the F1 score. The **macro F1** calculates the F1 scores of each class independently and reports the average of all classes. Since the ChEBI dataset is highly imbalanced – some classes have more than 100,000 positive samples while many are close to the threshold of 25 samples – the micro F1 mainly reflects performance on large classes while the macro F1 is determined by performance on smaller classes. A user study reported in [12] has determined that the correct prediction of smaller, more specific classes is particularly relevant to experts. Therefore, the evaluation in this work will focus on the macro F1. Differences between scores are reported in percentage points (pp).

The second metric is the consistency rate. It measures whether the prediction for a molecule is consistent with the constraints discussed in Section 3.2. While a perfect F1 score does guarantee near-perfect consistency, in practice this is rarely achievable. [3] Thus, for imperfect predictions, one needs to distinguish between inconsistent and consistent ones, the consistent ones being more useful in practice.

5.2 Base Learners

Before evaluating the ensemble aggregation in Section 5.3, we first take a closer look at the trained base learners. For an independent evaluation of each base learner, we refer to [50] for the GNNs, to [7] for ChemLog and to [10] for CCPs. The Lopster ruleset is evaluated in Appendix D. All base learners have been calibrated individually for their optimal threshold on the validation set. These thresholds will be used for all subsequent evaluations.

[3] In ChEBI version 252, which is used in this work, we identified nine molecules that had an inconsistent classification. These inconsistencies have since been resolved in ChEBI (as of version 255).

Table 15: Base learner performance on the test split of $\mathbf{ChEBI^{1,766}_{3-STAR}}$. #Labels is the number of labels the model can make predictions for in this subset. "missing excluded" excludes (molecule, label) pairs for which a model did not produce a prediction, "missing = negative" counts them as 0. F1 scores and consistency are given in percent.

		missing excluded		missing = negative		Consis-
Base learner	#Labels	micro F1	macro F1	micro F1	macro F1	tency
GAT-aug (3-STAR)	1,766	**91.85**	74.82	**91.85**	**74.82**	68.89
GAT (3-STAR)	1,766	90.98	73.76	90.98	73.76	60.17
RGGCN-aug (3-STAR)	1,766	91.44	73.71	91.44	73.71	67.54
RGGCN (3-STAR)	1,766	90.89	73.26	90.89	73.26	60.16
GAT-aug	1,766	89.19	73.30	89.19	73.30	62.98
GAT	1,766	86.69	67.78	86.69	67.78	48.18
RGGCN-aug	1,766	89.12	71.19	89.12	71.19	64.51
RGGCN	1,766	86.70	66.83	86.70	66.83	50.70
Lopster	55	82.88	**81.81**	51.14	2.20	**100.00**
CCPs	681	73.43	61.16	60.20	18.80	80.52
ChemLog	102	80.03	77.99	52.03	3.84	**100.00**

Table 14 shows performance results for the 11 base learners. Each base learner is evaluated both on the label subset it actually covers ("missing excluded") and the full label set ("missing=negative"). The strongest effect of the different evaluation methods can be seen for the rule-based models: The macro F1 of Lopster goes down from 74.94% to 1.8%. This indicates that the rules of Lopster are (relatively) accurate where they are applicable. But across the whole dataset, Lopster only covers a small portion with these rules.

Due to the different label coverages, the scores of different base learners are not fully comparable. One mitigation is an evaluation on the $\mathbf{ChEBI^{1,766}_{3-STAR}}$ subset (cf. Table 15), where DL models predict all labels, making those numbers comparable.

On $\mathbf{ChEBI^{1,766}_{3-STAR}}$, GAT-aug (3-STAR) is the best-performing DL model. In general, the augmented models tend to outperform the non-augmented ones and GAT tends to outperform RGGCN, with GAT-aug (3-STAR) beating RGGCN (3-STAR) by 1.56 pp in the macro- and 0.96 pp in the micro-F1. Regarding consistency, GAT and RGGCN perform similarly, with the main distinction lying between augmented and non-augmented models. The non-augmented models trained on $\mathbf{ChEBI^{2,117}_{full}}$ struggle the most: GAT only makes consistent predictions for 47.5% of inputs (48.18% in $\mathbf{ChEBI^{1,766}_{3-STAR}}$). Even compared on the same label set, GAT is substantially less consistent when trained on $\mathbf{ChEBI^{2,117}_{full}}$ compared to the same model trained on $\mathbf{ChEBI^{1,766}_{3-STAR}}$.

The symbolic models receive higher consistency values, with Lopster and ChemLog receiving 100% since they are consistent with the ChEBI axiomatisation by design. While the different coverage makes the F1 scores hard to compare, all models receive higher macro F1 scores on the $\mathbf{ChEBI^{1,766}_{3-STAR}}$ subset when excluding missing predictions (especially ChemLog, which gains 29.67 pp). This shows that some of the

low-scoring classes are classes which have fewer than 25 verified molecules in ChEBI. For ChemLog, these are mostly classes defined by an element that is relatively rare (in organic chemistry) and where not all molecules containing this element have the formal relation to their *molecular entity* class. Thus, these classes receive particularly low precision values (the macro precision of ChemLog is 43.7%). However, as discussed in [8], this low precision does not necessarily indicate that the rules are incorrect. Instead, it often points to ChEBI classes where the annotation is incomplete. Mitigating this issue is a task for future work.

5.3 Aggregation and Inconsistency Resolution

This evaluation compares the ensemble aggregation methods and inconsistency resolution algorithms discussed in Section 3. All algorithms have been calibrated / trained on the validation set. The identified parameters are $\alpha = 0.75, \beta = 8$ for WMV-F1 (trust, cf. Equation 9). DES uses the confidence-based vote $\mathbf{p}^{\text{des-conf}}$, region size $K = 1$, profile size $K_p = 9$ and decision threshold $\theta^{\text{des}} = 0.28$. This means that the region of competence in the feature space is reduced to the single most similar validation molecule and DES relies mainly on the neighbourhood in the output space (i.e., samples that have received similar base learner predictions). This indicates that neighbourhood in the feature space did not provide the model with useful information. For LTR, the selected parameters are $k = 70$ (candidate set size per base learner), $a = 2.23$ and $b = -1.62$ (Platt scaling parameters) and $\theta^{\text{ltr}} = 0.06$. The full model configurations can be found in the Supplementary Material.

Table 16 shows the grid of aggregation (rows) and inconsistency resolution (columns) combinations. As for the base learners, the F1 score is reported as both micro- and macro-aggregated on both $\mathbf{ChEBI}^{\mathbf{2,234}}_{\mathbf{3-STAR}}$ and $\mathbf{ChEBI}^{\mathbf{1,766}}_{\mathbf{3-STAR}}$. The latter is relevant for comparing the ensemble results to the base learners, as no single base learner can make predictions for all classes. In addition, we also report the sample-wise consistency.

The first result to highlight is that all ensemble methods except for LTR beat the best DL models substantially. On $\mathbf{ChEBI}^{\mathbf{1,766}}_{\mathbf{3-STAR}}$, WMV-F1 with 3-step / ILR-Łukasiewicz resolution achieves a micro F1 of 93.44% (+1.59 pp compared to GAT-aug 3-STAR). DES with ILR-Łukasiewicz achieves a macro F1 of 80.71% (+5.89 pp). The simplest ensemble alone (MV without inconsistency resolution) reaches a macro F1 3.43 pp above GAT-aug (3-STAR). This shows that, by simply combining the base learners, the overall predictions can be improved. The same goes for consistency, where MV reaches 79.65% (+10.76 pp) and WMV-Conf reaches 83.29%. Even without explicit inconsistency resolution, roughly a third of the inconsistent predictions can be repaired by a majority vote.

Taking a closer look at the voting methods, a clear order can be seen: WMV-Conf improves upon MV, and WMV-F1 improves upon WMV-Conf. This is in line with our expectations as each method adds another source of information to the ensemble: MV decides by the majority of models, WMV-Conf weights each model by its confidence and WMV-F1 adds the F1 score-based trust to the model confidence. DES achieves the highest macro F1, but a lower micro F1 than the voting methods. This indicates that it is particularly good at predicting a range of smaller classes, but struggles on some of

Table 16: F1 score / consistency (in percent) of aggregation-inconsistency resolution combinations on the test split.

Metric	Aggregation	none	3-step	ILR-Gödel	ILR-Łukasiewicz	HEX
micro-F1 $\mathbf{ChEBI^{2,234}_{3-STAR}}$	MV	92.77	92.83	92.69	92.81	91.99
	WMV-Conf	92.99	92.97	92.89	92.96	92.29
	WMV-F1	93.28	**93.32**	93.06	**93.32**	92.94
	DES	92.42	92.80	91.35	92.58	92.24
	LTR	88.67	89.44	87.19	88.18	91.43
micro-F1 $\mathbf{ChEBI^{1,766}_{3-STAR}}$	MV	92.89	92.95	92.81	92.93	92.10
	WMV-Conf	93.11	93.08	93.01	93.08	92.40
	WMV-F1	93.41	**93.44**	93.19	**93.44**	93.05
	DES	92.56	92.92	91.48	92.71	92.35
	LTR	88.86	89.57	87.36	88.33	91.53
macro-F1 $\mathbf{ChEBI^{2,234}_{3-STAR}}$	MV	69.58	69.65	69.60	69.59	67.99
	WMV-Conf	70.80	70.63	70.79	70.59	68.96
	WMV-F1	71.46	71.64	71.47	71.57	70.59
	DES	71.89	**72.44**	71.60	**72.44**	69.24
	LTR	67.42	70.00	66.63	68.18	69.89
macro-F1 $\mathbf{ChEBI^{1,766}_{3-STAR}}$	MV	78.25	78.33	78.30	78.26	76.34
	WMV-Conf	79.27	79.07	79.26	79.03	77.26
	WMV-F1	79.91	80.03	79.94	79.92	78.96
	DES	80.02	80.62	79.65	**80.71**	77.64
	LTR	75.07	77.52	73.94	75.44	78.71
Consistency	MV	79.65				
	WMV-Conf	**83.29**				
	WMV-F1	79.10			100	
	DES	67.58				
	LTR	66.62				

the larger classes. One possible explanation for this behaviour lies in its local approach: DES works independently for every label, taking the output space neighbourhood as its basis. This can be very efficient when having to make the most of a limited number of positive samples, but can also fail at getting the full picture of larger classes. In comparison, WMV-F1 might get imprecise weights for small classes where the F1 score is calculated based on only a handful of samples, but the weights become more meaningful for larger classes. Remarkably, despite the good macro F1 score, in terms of consistency, DES cannot compete with the voting methods, leaving roughly twice as many molecules inconsistent as WMV-Conf. This could be a side effect of DES making (correct) predictions for classes at the bottom of the label hierarchy while in some cases missing classes further upwards. LTR is the worst-performing of our selection of methods. Here, the striking difference to the rest is the number of predictions made (163,666 positive predictions across the test set, compared to 144,908 for DES, both without resolution). Consequently, the balance between recall and precision is shifted: While LTR reaches the highest recall of all aggregations (81.4%, against 76.2% for DES), it underperforms on precision (60.8% against 71.1% for DES).

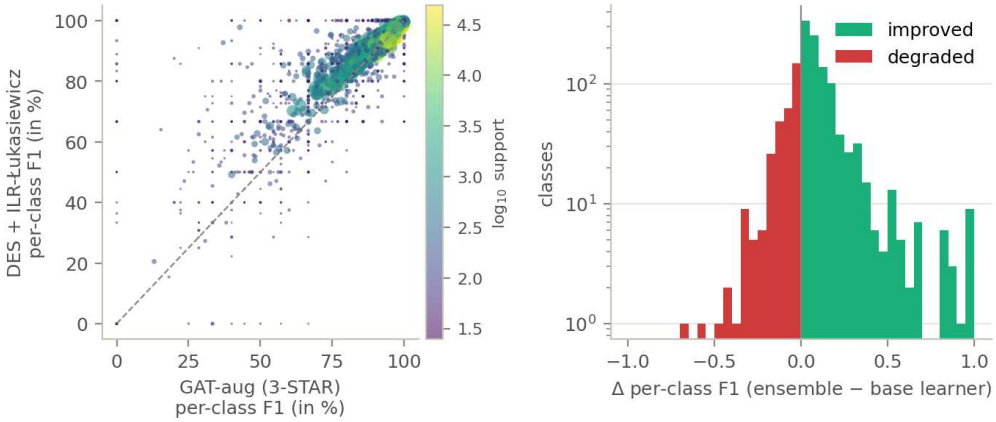


Fig. 7: Class-wise comparison between the best base learner (GAT-aug 3-STAR) and the best ensemble (DES with ILR-Łukasiewicz) on the $\mathbf{ChEBI}^{1,766}_{3-\mathbf{STAR}}$ test set. Left: Each point represents a class, with classes with more positive samples being represented by larger points.

The inconsistency resolution methods show a clear pattern across ensemble configurations. 3-step and ILR-Łukasiewicz produce near-identical results, both slightly improving on the no-resolution F1 scores. This shows that both methods behave very similarly in practice. Both give priority to confident predictions while predictions close to the decision threshold get pulled to the other side if necessary. In contrast, ILR-Gödel is more "optimistic", only correcting subsumption violations upwards. This is reflected in the macro precision and recall: For DES, ILR-Gödel increases recall by 0.6 pp, but decreases precision by 1 pp. HEX follows the opposite strategy, most often correcting predictions downwards. This lowers recall by 8.6 pp, but increases precision by 4.1 pp. While the "balanced" strategy of 3-step / ILR-Łukasiewicz seems to be the best choice for most cases, LTR benefits most from HEX. This matches the earlier analysis that LTR is overly optimistic. Here, HEX helps counteract LTR's tendency and corrects a large number of false positives. Another noteworthy observation is that DES gains more from inconsistency resolution than the voting methods. With ILR-Łukasiewicz, the macro F1 of WMV-F1 increases by 0.11 pp / 0.01 pp (on $\mathbf{ChEBI}^{2,234}_{3-\mathbf{STAR}}$/ $\mathbf{ChEBI}^{1,766}_{3-\mathbf{STAR}}$), but the macro F1 of DES increases by 0.55 pp / 0.69 pp. Apparently, given more inconsistent relations, 3-step and ILR-Łukasiewicz are not only able to repair, but also improve the original predictions.

5.4 Class-wise Analysis

For a closer analysis of the results, we break down the prediction results by class. For which classes does the ensemble improve against the base learners? Where does it underperform? In the following, DES with ILR-Łukasiewicz is used as the ensemble with the highest macro F1 and GAT-aug (3-STAR) is used as the best DL base learner. Consequently, the evaluation is restricted to $\mathbf{ChEBI}^{1,766}_{3-\mathbf{STAR}}$.

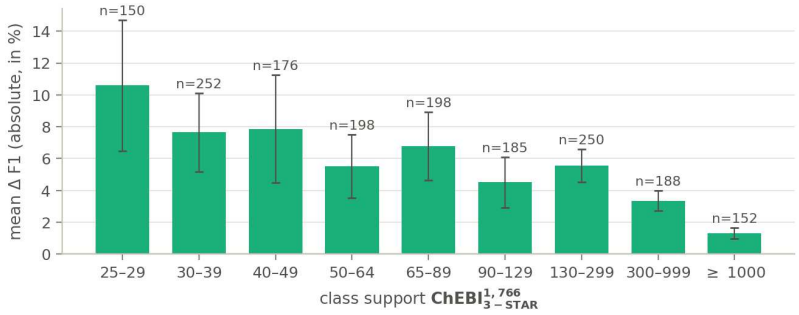


Fig. 8: Absolute performance gain (F1 score) of the best ensemble (DES with ILR-Łukasiewicz) vs. the best base learner (GAT-aug 3-STAR), binned by class support and shown with 95% confidence intervals.

Figure 7 shows the performance of the ensemble and the base learner per class. Most classes perform similarly for both, with a slight tendency towards the ensemble. In total, the ensemble improves upon the base learner for 1,000 classes and degrades for 310 (the remaining classes are tied). While outliers with a large difference between ensemble and base learner exist (mostly in the ensemble direction), those mostly represent smaller classes. The bulk of classes (especially larger ones) shows a strong correlation between base learner and ensemble F1 score, with only a small delta.

This impression is confirmed by Figure 8, which bins the classes by their support (i.e., the number of positive samples across the dataset). Here, a clear tendency is visible where smaller classes have higher gains (> 10 pp) while the largest classes stay nearly unchanged (< 2 pp). There are two factors driving this pattern: Firstly, larger classes tend to have less room for improvement. Most of them have already been learned near-perfectly by the base learners. Secondly, the different base learners complement each other. While each DL model fails at some molecules or classes, not all models fail at the same molecules or classes. Thus, although GAT-aug (3-STAR) is the strongest DL model overall, for some classes, it is overruled by the other DL models. In addition, the symbolic models can provide their own contributions for selected classes. Their advantage (besides the coverage of classes outside the DL training set, which is not shown in this section) is that they are not data-dependent. So even if a class is difficult to learn (for instance, because not enough data points are available or the data points are noisy), they can still make accurate predictions.

Instead of by size, we can also group ChEBI classes by chemical features. This grouping is based on regex patterns on the class names (see Appendix E). Table 17 shows the F1 scores and numbers of improved / degraded classes for 14 groups. The positive outliers are alkaloids and N-heterocycles, where the ensemble was able to beat GAT-aug (3-STAR) by more than 15 pp on average. At the other end of the spectrum, CoA is the only group where the ensemble loses for the majority of classes. This shows that the ensemble gains are not equally distributed across the class spectrum. Groups where the base learner has a low F1 score benefit from the ensemble in particular ($47.91\% \rightarrow 71.36\%$ for alkaloids). Other groups are more diverse. For instance, 92 sugar classes improved with the ensemble, sometimes by a large margin (e.g., *TDP*, *dTDP*

Table 17: Macro F1 and number of improved / degraded labels for label subsets, comparing the best base learner (GAT-aug 3-STAR) with the ensemble (DES with ILR-Łukasiewicz) on the $\mathbf{ChEBI}^{\mathbf{1,766}}_{\mathbf{3-STAR}}$ test set.

		Macro F1 (%)			Classes	
Label group	n	Ensemble	Base learner	Δ (pp)	Impr.	Degr.
Alkaloid	18	71.36	47.91	23.45	12	2
N-heterocycle	136	80.63	64.06	16.57	99	5
Terpenoid	46	86.91	77.97	8.93	32	5
Sugar/glycoside	172	78.11	70.78	7.33	92	42
Peptide	19	82.71	75.61	7.10	16	2
Metal/element entity	89	87.43	81.18	6.24	44	27
Fatty acid / acyl anion	176	80.96	76.27	4.69	85	31
Amino-acid derivative	79	74.17	69.61	4.57	44	14
Flavonoid	46	85.92	83.26	2.66	22	11
Glycerophospholipid/acylglycerol	78	88.39	85.85	2.54	29	8
Ceramide/sphingolipid	47	87.70	85.61	2.09	14	8
S/Se-containing	77	82.79	80.80	1.99	31	26
Steroid	77	84.73	83.77	0.96	30	21
CoA/acyl-CoA	45	80.96	81.53	-0.57	15	18
Any family	984	82.22	76.10	6.12	516	190
No family	782	80.57	74.85	5.72	484	120

and *UDP-sugar* each gained more than 80 pp, cf. Appendix F). However, 42 classes degraded as well. Overall, the ensemble acts as a stabiliser: every group reaches a macro-F1 $\geq 70\%$ (with the base learner, three groups remain below 70%). Apparently, weaknesses of base learners in specific areas can be compensated for by the ensemble aggregation. The downside is that, in some areas (e.g., CoA), it is also possible for a base learner to get drowned out by other, weaker base learners.

5.5 Base Learner Attribution

Until now, the evaluation has shown that the ensemble methods improve predictions, especially for small classes. However, this raises the question of which base learners are actually responsible for this. That is, do all of the base learners contribute equally to the ensemble? And what happens if some of the base learners get removed? In this section, the WMV-F1 ensemble with 3-step inconsistency resolution is used. While this configuration does not have the highest macro F1, it has the highest micro F1 and, more importantly, does not rely on a trained model. This makes base learner attributions easier.

To measure the contribution of a base learner to a prediction, the weights of the WMV-F1 ensemble are measured directly for every (molecule, label, base learner) tuple. By normalising against the sum of weights for each (molecule, label) pair, we get a relative score for every base learner. Since the final predictions can be changed by inconsistency resolution, we trace the flow of predictions through the 3-step resolution algorithm: If the prediction of a class A is changed due to the relation with another class B, the attribution of B gets propagated to A. For instance, consider a class A that

Table 18: Directly measured impact of each base learner on the ensemble result (WMV-F1 + 3-step, $\mathbf{ChEBI}^{\mathbf{2,234}}_{\mathbf{3-STAR}}$ test set). Mean$^+$ is the average on positive predictions only. "Improved labels" refers to labels where the ensemble has a higher F1 score than GAT-aug (3-STAR).

	all labels			improved labels	
Base learner	mean	mean$^+$	max	mean	Δ (pp)
GAT-aug (3-STAR)	11.69	**13.81**	98.22	13.77	+2.07
GAT (3-STAR)	10.47	12.42	88.90	13.23	+2.76
RGGCN-aug (3-STAR)	11.18	13.59	96.85	**14.11**	+2.93
RGGCN (3-STAR)	10.61	12.90	93.57	13.99	+3.38
GAT-aug	**14.41**	11.53	99.94	12.76	-1.65
GAT	11.24	8.96	99.92	10.04	-1.20
RGGCN-aug	12.65	11.08	100.00	11.44	-1.21
RGGCN	10.99	9.19	100.00	9.14	-1.85
Lopster	0.30	2.04	100.00	0.21	-0.09
CCPs	1.17	1.89	100.00	0.76	-0.40
ChemLog	5.28	2.58	100.00	0.55	-4.74

is a superclass of B, and base learners m and n with predictions $p^m_A(x) = 0.55, p^m_B(x) = 0.35$ and $p^n_A(x) = 0.4, p^n_B(x) = 0.95$. For simplicity, we assume $\theta^m = \theta^n = 0.5$ and a trust of 1 for all base learners and labels. Then, the WMV-F1 ensemble calculates its predictions as

$$p^e_A(x) = \frac{\text{conf}^m_A(x)}{\text{conf}^m_A(x) + \text{conf}^n_A(x)} = \frac{0.1}{0.1 + 0.2} = \frac{1}{3} \tag{22}$$

and

$$p^e_B(x) = \frac{\text{conf}^n_B(x)}{\text{conf}^m_B(x) + \text{conf}^n_B(x)} = \frac{0.9}{0.3 + 0.9} = \frac{3}{4} \tag{23}$$

The contributions for A are $\frac{1}{3}$ / $\frac{2}{3}$ (m / n) and $\frac{1}{4}$ / $\frac{3}{4}$ for B. Since the predictions are inconsistent, the resolution algorithm corrects the prediction of A upwards. Thus, we report $\frac{1}{4}$ / $\frac{3}{4}$ as the contributions for both A and B. Intuitively put: n has the main responsibility for both predictions by confidently predicting B, and m only has a minor contribution by not rejecting B confidently enough (and not being confident enough in the prediction of A).

Table 18 and Figure 9 show the attributions for each base learner. "mean" and "mean$^+$" are the average over all predictions and positive predictions, respectively. "max" is the maximum attribution for a label. For labels where the ensemble improved upon GAT-aug (3-STAR), we report the mean attribution and the difference to the overall attribution.

Overall, attributions are highly distributed, with each DL model contributing at least 10%. With 14.41%, GAT-aug has the highest mean attribution. This is expected since it covers more labels than the 3-STAR DL models. The latter win when looking only at positive predictions. Here, the extended coverage is less important (as it only affects rare classes). Within the DL models, the order follows the F1 scores reported in Section 5.2: Augmented models get a higher share in decisions than non-augmented

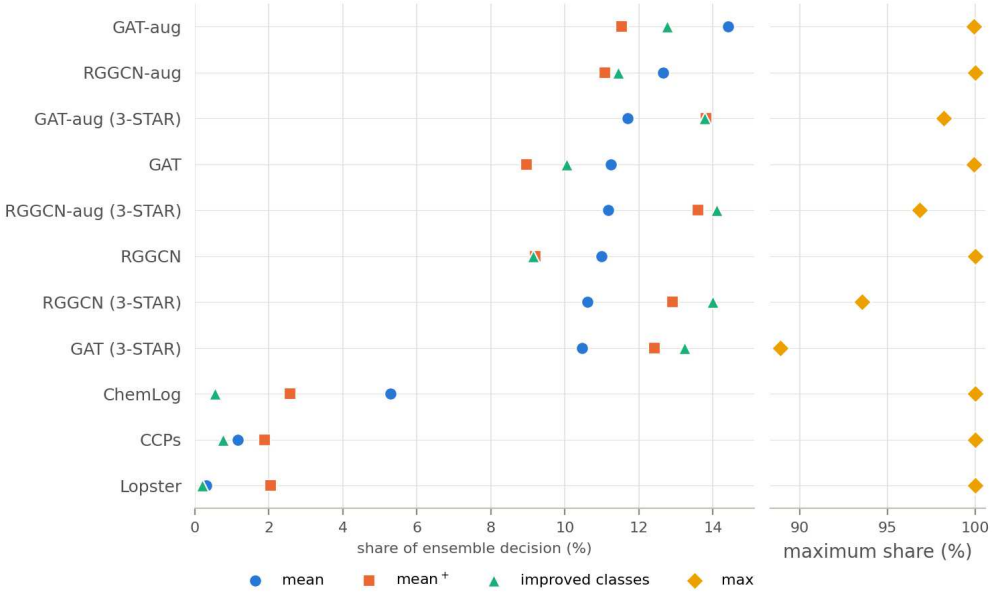


Fig. 9: Base learner attribution for the WMV-F1 ensemble with 3-step inconsistency resolution (cf. Table 18).

ones. This correlation is trivial since the F1 score is directly used as the ensemble weight. The symbolic models have a lower impact on the predictions, which can be explained by their lower label coverage. Nevertheless, they have a maximum impact of 100% since each symbolic model covers labels that no other base learner predicts. The DL models also have a high maximum impact, especially the full-ChEBI models, which all get at least 99.9% of the share on at least one label. This means that each DL model almost unilaterally decides on at least one label.

A considerable shift happens when only taking classes into account where the ensemble performed better than the best base learner, GAT-aug (3-STAR), excluding classes where GAT-aug (3-STAR) cannot make predictions. Here, the 3-STAR models have a higher impact, while all other models lose impact. Especially the contribution of ChemLog diminishes to 0.55%. One can interpret this as ensemble gains coming mostly from the interaction between 3-STAR and (to a slightly lesser extent) full-ChEBI DL models. Symbolic models do not seem to play a major role here. Instead, symbolic models provide two contributions to the ensemble: firstly, the extended class coverage, and secondly, performance gains that are invisible when measuring only the F1 score. As discussed in Section 5.2, the rules for Lopster and ChemLog have been manually crafted and verified against ChEBI data. Thus, we argue that these rules can be trusted and should stay part of the ensemble even if they do not improve the F1 score. This is mainly a limitation of our metrics and the underlying dataset and not one of the symbolic models.

While Table 18 shows that all DL models contribute to the ensemble, this does not answer the question of whether all models are strictly necessary for the ensemble

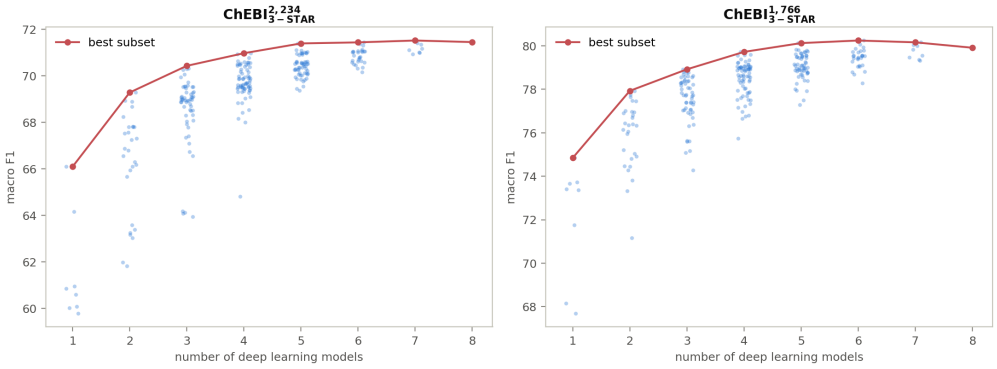


Fig. 10: Macro F1 for subsets of ensemble base learners (aggregated with WMV-F1, no inconsistency resolution). All subsets include the symbolic models and a given number of DL models. Evaluated on the test set with $\mathbf{ChEBI^{2,234}_{3-STAR}}$ (left) and $\mathbf{ChEBI^{1,766}_{3-STAR}}$ (right) labels.

to function. That is, when removing one model, would the remaining ones be able to pick up the slack? To address this question, we have evaluated the ensemble (WMV-F1, for performance reasons without inconsistency resolution) for 255 different sets of base learners ranging from one DL model to all DL models (symbolic models are always included).

The results are displayed in Figure 10. The macro F1 score increases up to an ensemble size of five DL models. Adding more models does not substantially improve performance beyond that point. In fact, for classes in the $\mathbf{ChEBI^{1,766}_{3-STAR}}$ subset, it can even slightly reduce the macro F1 score. The choice of which DL models to select at a given ensemble size is also relevant. Especially for smaller ensemble sizes, the spread between the best- and worst-performing subsets increases. Overall, the best ensemble subset on $\mathbf{ChEBI^{1,766}_{3-STAR}}$ is the 6-DL configuration with all 3-STAR models, GAT-aug and GAT (not using RGGCN and RGGCN-aug), reaching a macro F1 score of 80.24% (+0.33 pp vs. 8-DL).

In addition, we have calculated the Shapley values for each DL base learner from the ensemble subsets (cf. Table 19). The Shapley value of a base learner is its average marginal contribution, i.e., the macro F1 gain obtained by adding it to a subset of the other DL models, averaged over all such subsets with weights that give each subset size equal influence. The values sum to the gain of the full ensemble over the symbolic models alone, that is, roughly 53 and 59 percentage points on the two label sets. These values are again in line with the attributions measured in Table 18: All models receive similar values, with better-performing models getting slightly higher ones. The pattern of full-ChEBI models contributing more to $\mathbf{ChEBI^{2,234}_{3-STAR}}$ and 3-STAR models contributing more to $\mathbf{ChEBI^{1,766}_{3-STAR}}$ is also visible. This underlines the conclusion that all DL models contribute to the ensemble, but some are replaceable without losing performance. Only when more than three models are removed does ensemble performance degrade noticeably.

Table 19: Shapley values ϕ (in percentage points of macro F1) of each DL base learner, i.e., the weighted average of the macro F1 gain obtained by adding the base learner to each subset of the other DL models. The symbolic models are always included. By construction, the values sum to the gain of the full ensemble over the symbolic models alone.

Base learner	ϕ $\mathbf{ChEBI}^{\mathbf{2,234}}_{\mathbf{3-STAR}}$	ϕ $\mathbf{ChEBI}^{\mathbf{1,766}}_{\mathbf{3-STAR}}$
GAT-aug (3-STAR)	+6.59	**+8.34**
GAT (3-STAR)	+6.18	+7.82
RGGCN-aug (3-STAR)	+6.38	+8.07
RGGCN (3-STAR)	+6.13	+7.76
GAT-aug	**+7.95**	+7.47
GAT	+6.30	+6.26
RGGCN-aug	+7.10	+6.86
RGGCN	+6.22	+6.08
sum	+52.85	+58.65

6 Conclusion

In this work, we have presented Chebifier 2, an ensemble framework for the ChEBI ontology extension task. The ensemble combines eight deep learning and three symbolic classifiers into a single prediction over 2,234 ChEBI classes. Ensemble methods in the literature have mostly been developed for binary and multi-class problems. We have extended the methodology, both for aggregation and for inconsistency resolution, to the hierarchical multi-label setting that arises naturally in ontology extension tasks, where the label space is structured by subsumption and disjointness axioms.

We have studied five aggregation methods. The voting methods majority voting (MV) and the two weighted majority voting schemes WMV-Conf and WMV-F1 provide increasing performance by adding further information: the confidence of the individual prediction and the validation performance of the base learner. Among the meta-learning methods, dynamic ensemble selection (DES) achieves the highest macro F1, while learning to rank (LTR) falls behind, as it produces too many positive predictions. Since the ChEBI ontology provides subsumption and disjointness axioms, we let an inconsistency resolution step follow the aggregation. We have compared three resolution methods, Hierarchy and Exclusion (HEX) graphs, Iterative Local Refinement (ILR, based on a fuzzy logic such as Gödel logic or Łukasiewicz logic) and a novel 3-step algorithm, and adapted them to the multi-label case. All three remove every inconsistency, but they differ in character and in resulting predictive performance: 3-step and ILR-Łukasiewicz behave almost identically and balance corrections in both directions, ILR-Gödel is optimistic and corrects violations upwards, and HEX is pessimistic, corrects predictions downwards and performs worst overall. The best combinations are DES with ILR-Łukasiewicz, reaching a macro F1 of 80.7% on the $\mathbf{ChEBI}^{\mathbf{1,766}}_{\mathbf{3-STAR}}$ labels, 5.9 percentage points above the best single classifier, and WMV-F1 with 3-step

resolution, reaching the highest micro F1 of 93.4%. Even the simplest ensemble, majority voting without inconsistency resolution, improves the macro F1 by 3.4 percentage points.

The gains are not distributed evenly. They are largest for small, chemically difficult (and interesting) classes, where the ensemble improves upon the best base learner by more than 10 percentage points, while the largest classes remain nearly unchanged. Most deep learning base learners contribute substantially to the ensemble, each receiving at least 10% of the attributed decisions, but they are also largely interchangeable: five of the eight models suffice, and the Shapley values of all models lie in a narrow range. The symbolic classifiers barely affect the F1 score, but they extend the coverage of the ensemble by 117 classes for which insufficient training data (fewer than 25 samples) is available, are consistent by construction, and rest on rules that have been verified by experts. Their comparatively low precision frequently points to incomplete annotations in ChEBI rather than to errors in the rules.

Our evaluation has limitations. It is restricted to ChEBI, and it uses ChEBI itself as the gold standard, although its annotations are known to be incomplete. Future work will therefore include feeding the annotation gaps uncovered by the symbolic classifiers back into ChEBI. Further directions are the transfer of the methodology to other ontologies with subsumption and disjointness axioms, such as the Gene Ontology from which LTR originates; the integration of new and improved base learners, for instance classifiers based on inductive logic programming (ILP) [68] and/or LLMs; the combination of the strengths of DES on small classes and WMV-F1 on large classes; and explanations for ensemble decisions that trace a prediction through aggregation and inconsistency resolution, which we have deliberately left out of the scope of this work.

Chebifier 2 is available as a Python library [18] and as a web frontend [19].

Supplementary information. The implementation of our ensemble can be found at https://github.com/ChEB-AI/python-chebifier. An interactive website for chemical classification can be found at https://chebifier.hastingslab.org/. The website also provides explanations of individual classifications for ChemLog and CCPs. Trained models and ensemble weights are available at https://zenodo.org/records/23161410.

Declarations

6.1 Availability of Data and Materials

Supplementary data is available at https://zenodo.org/records/23161410.

The following repositories were used in this work:

- `chebifier` (https://github.com/ChEB-AI/python-chebifier, v1.3.0) - Implementation of the ensemble architecture and bindings for the base learner models.
- `chebifier-web` (https://github.com/ChEB-AI/chebifier-web) - Web frontend for *chebifier*, running at https://chebifier.hastingslab.org/
- `chebai` (https://github.com/ChEB-AI/python-chebai, v1.4.0) - DL library used to train and evaluate DL models on the ChEBI dataset

- `chebai-graph` (https://github.com/ChEB-AI/python-chebai-graph, v1.2.3) - Extension of *chebai* to GNNs
- `c3p` (https://github.com/chemkg/c3p, v0.5.0) - Repository for the Chemical Classifier Programs
- `chemlog` (https://github.com/sfluegel05/chemlog-peptides, v1.0.7) - Molecule classification based on rules, covers peptide rules [8] and Lopster rules [9]
- `chemlog-extra` (https://github.com/ChEB-AI/chemlog-extra, v1.1) - Extension of *chemlog* to other ChEBI classes

6.2 Competing Interests

The authors have no relevant financial or non-financial interests to disclose.

6.3 Funding

This work has been funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) – 522907718 and 456666331 and by the Swiss National Science Foundation (SNF) – 215906.

6.4 Author contributions

Conceptualisation: TM, JH, FN; Methodology: SF, MG; Formal analysis and investigation: SF, MG; Software: SF, AGK; Writing – original draft preparation: SF; Writing – review and editing: SF, MG, JH, TM, CT, FN, AGK; Funding acquisition: TM, JH; Supervision: TM, JH, FN.

6.5 Acknowledgements

Not applicable.

Appendix A HEX Graph Bounded Inference

A central problem of the HEX graph inconsistency resolution (cf. Section 3.2.1) is that its performance depends on the structure of the label space. Calculating the predictions requires iteration over the set of consistent assignments Y^e. According to Theorem 3 of [16], the number of consistent assignments is bounded by

$$|Y^e| \leq (|V| - \Omega_G + 1)2^{\Omega_G} \tag{A1}$$

where Ω_G is the maximum overlap of G, i.e., the highest number of classes with which some node of G does not share an edge in G. The algorithm suggested by [16] takes this issue into account and splits G into cliques by forming a junction tree. In the inference algorithm, the state space is enumerated per clique and sum-product message passing is used to align the cliques. This brings inference complexity down to $O(\min\{|V|2^w, |V|^2 2^{\Omega_G}\})$ (Theorem 4 of [16]) given the junction tree width w. However, this is not sufficient for our use case. When constructed for the ChEBI labels, the junction tree width is 62 and the number of possible assignments for the largest clique

alone is $> 2^{30}$. This makes neither assignment enumeration for the individual cliques nor message passing between the cliques viable.

The underlying reason for this lies in the structure of our label hierarchy. While Deng et al. [16] evaluate their method on a multi-class problem and assume all leaf classes to be disjoint, in this work, the problem is a multi-label one where multiple assigned leaf classes are a common feature. Actual disjointness relations are comparatively sparse and have to be defined explicitly. Therefore, the assignment space grows exponentially for ChEBI.

Algorithm 2 HEX-bounded inference algorithm

Input: HEX Graph $G = (V, E_h, E_e)$
Input: logits $\mathbf{l}^e \in \mathbb{R}^C$
Input: maximum number of iterations n
Input: set of classes C
Output: $\mathbf{p^{hex-bounded}}(x) \in [0,1]^C$
$a_0 \leftarrow \textsc{Close}(\emptyset, \emptyset)$
$\mathcal{A} \leftarrow \{a_0\}$ ▷ Initialise frontier
for all $i \in \{0, 1, \ldots, n\}$ **do**
  $a^* = (P^*, Q^*) \leftarrow \arg\max_{a \in \mathcal{A}}(U(a) - L(a))$
  $k^* \leftarrow \arg\max_{k \in V \setminus (P^* \cup Q^*)} \text{softplus}(l^e_k)$
  $\mathcal{A} \leftarrow (\mathcal{A} \setminus \{a^*\}) \cup \{\textsc{Close}(P^* \cup \{k^*\}, Q^*), \textsc{Close}(P^*, Q^* \cup \{k^*\})\}$
  for all $c \in C$ **do**
    $\pi^{lo}_c, \pi^{hi}_c \leftarrow \textsc{Aggregate}(\mathcal{A}, c)$
  end for
  if $\forall c \in C : (\pi^{lo}_c > \theta^{hex} \Leftrightarrow \pi^{hi}_c > \theta^{hex})$ **then** ▷ Early stopping
    BREAK
  end if
end for
$\mathbf{p}^{\text{hex-bounded}}(x) \leftarrow \pi^{\mathbf{lo}}$

To address this, we propose a new inference algorithm (cf. Algorithm 2) that does not enumerate the whole assignment space. Instead, it approximates $p^{\text{hex}}_c(x)$ by defining lower and upper bounds that are iteratively refined. Each iteration partitions the assignment space into segments for which the lower and upper bounds can be tightened effectively. Each segment is a *partial assignment* $a = (P, Q)$ that defines the subset of assignments setting all labels in P as positive and all labels in Q as negative:

$$Y(a) = \{\mathbf{y^e} \in Y^e | y_P \equiv 1 \wedge y_Q \equiv 0\} \tag{A2}$$

The weight of a partial assignment is the weight of its assignments:

$$W(a) = \sum_{\mathbf{y^e} \in Y(a)} w(\mathbf{y^e}) \tag{A3}$$

A partial assignment is *closed* if P contains all superclasses of labels in P and Q contains all subclasses of labels in Q as well as all labels that are disjoint with a label in P. Since the hierarchy edges E_h already include the transitive closure of the subsumption relation and the exclusion edges E_e already include all disjointness relations propagated to subclasses, the closure of a partial assignment can be inferred trivially (cf. Algorithm 3).

Algorithm 3 Partial assignment closure for HEX-bounded inference

```
function CLOSE(P, Q)
    P' ← P ∪ {j | (v_j, v_i) ∈ E_h for some i ∈ P}
    Q' ← Q∪{j | (v_i, v_j) ∈ E_h for some i ∈ Q}∪{j | (v_i, v_j) ∈ E_e for some i ∈ P}
    return P', Q'
end function
```

For a closed partial assignment a, we can calculate lower and upper bounds such that $L(a) \leq W(a) \leq U(a)$. $L(a)$ is the weight of the assignment with only the labels in P set as positive (so all undecided class labels are set as negative):

$$L(a) = \prod_{c \in P} e^{l_c^e(x)} \tag{A4}$$

The upper bound also takes the weights of the classes in P, but multiplies them by $1 + e^{l_c^e(x)}$ for each undecided class. This is the sum of both options: either the undecided class is negative (contributing a factor of 1) or it is positive (contributing its odds $e^{l_c^e(x)}$). The weight would reach this upper bound if all undecided classes had no constraints.

$$U(a) = \prod_{c \in P} e^{l_c^e(x)} \prod_{c \in V \setminus (P \cup Q)} (1 + e^{l_c^e(x)}) \tag{A5}$$

A *frontier* $\mathcal{A}$ is a set of closed partial assignments partitioning the state space:

$$Y^e = \biguplus_{a \in \mathcal{A}} Y(a) \tag{A6}$$

In Algorithm 2, it is initialised with the closure of $a = (\emptyset, \emptyset)$. At each iteration, we choose the partial assignment a^* with the largest uncertainty and split it into two halves along a label k^*. k^* is the free label in a^* with the highest contribution to this uncertainty. This produces two partial assignments, one with $k^* \in P^*$, the other with $k^* \in Q^*$, which are both added to $\mathcal{A}$.

The lower and upper bounds of the partial assignments are propagated to label bounds by Algorithm 4 such that

$$\pi_c^{lo} \leq p_c^{hex}(x) \leq \pi_c^{hi} \tag{A7}$$

While N_c^{hi}, M_c^{lo} and M_c^{hi} are direct sums of lower / upper bounds of partial assignments, N_c^{lo} is less intuitive. Here, for the partial assignments $\mathcal{A}^F$ where c is free, we

Algorithm 4 Frontier aggregation for HEX-bounded inference

function AGGREGATE($\mathcal{A}$, c) ▷ Propagate bounds to the labels
  $\mathcal{A}^P \leftarrow \{(P,Q) \in \mathcal{A} \mid c \in P\}$
  $\mathcal{A}^Q \leftarrow \{(P,Q) \in \mathcal{A} \mid c \in Q\}$
  $\mathcal{A}^F \leftarrow \mathcal{A} \setminus (\mathcal{A}^{\mathcal{P}} \cup \mathcal{A}^{\mathcal{Q}})$
  $\mathcal{A}^{Fc} \leftarrow \{\text{CLOSE}(P \cup \{c\}, Q) \mid (P,Q) \in \mathcal{A}^F\}$
  $N_c^{lo} \leftarrow \sum_{a \in \mathcal{A}^P} L(a) + \sum_{a \in \mathcal{A}^{Fc}} L(a)$
  $N_c^{hi} \leftarrow \sum_{a \notin \mathcal{A}^Q} U(a)$
  $M_c^{lo} \leftarrow \sum_{a \notin \mathcal{A}^P} L(a)$
  $M_c^{hi} \leftarrow \sum_{a \notin \mathcal{A}^P} U(a)$
  $\pi_c^{lo} \leftarrow \frac{N_c^{lo}}{N_c^{lo}+M_c^{hi}}$
  $\pi_c^{hi} \leftarrow \frac{N_c^{hi}}{N_c^{hi}+M_c^{lo}}$
  return π_c^{lo}, π_c^{hi}
end function

can add c to the assignment to increase the lower bound, creating a new set of partial assignments $\mathcal{A}^{Fc}$. This is possible because the partial assignments in $\mathcal{A}$ are closed. Therefore, if c is still free, it means that there is a possible assignment containing c as a positive label.

We call a label *certified* if the lower and upper bounds are on the same side of the decision threshold θ^e. If a label is certified, the ensemble's final decision for this label is necessarily the same as the one it would produce given full inference over the label space. If all labels are certified or the maximum number of iterations is reached (in this work, we use 2,000), the inference algorithm terminates. Preliminary experiments have shown that our algorithm certifies $> 99\%$ of labels across the test set, reaching certification on average after 73.7 iterations [4]. If a label is not certified, the lower bound is chosen. This ensures consistency since the lower bounds are monotone ($\pi_A^{lo} \leq \pi_B^{lo}$ for A subclass of B) and trivially satisfy the disjointness condition. This makes Algorithm 2 a suitable approximation of the inference algorithm proposed by [16].

Appendix B ILR Refinement Functions

This section covers the refinement functions used by the Iterative Local Refinement (ILR, cf. Section 3.2.2) algorithm. Based on the Gödel and Łukasiewicz t-norms, these are the minimal refinement functions needed to reach the desired target value. The formulas used in this work are special cases of the ones discussed in [17].

The Gödel refinement functions for subsumption / disjointness relations are

$$\rho^*_{R_G}(t_A, t_B, 1) = \begin{bmatrix} t_A \\ \max(t_A, t_B) \end{bmatrix} \tag{B8}$$

[4] In practice, the algorithm is run for at least 1,000 iterations since it is more computationally efficient than checking for convergence at every iteration.

and

$$\rho^*_{T_G}(t_A, t_B, 0) = \begin{bmatrix} \begin{cases} 0 & \text{if} \quad t_A \leq t_B \\ t_A & \text{otherwise} \end{cases} \\ \begin{cases} 0 & \text{if} \quad t_B < t_A \\ t_B & \text{otherwise} \end{cases} \end{bmatrix} \tag{B9}$$

For subsumptions, this means that the Gödel implication always leaves the subclass unchanged and bumps up the superclass. For disjointness relations, the Gödel t-norm sets the lower value to 0.

The Łukasiewicz implication R_L and t-norm T_L show a noticeably different behaviour. The Łukasiewicz implication is the fuzzy counterpart of $\neg A \vee B$: $R_L(t_A, t_B) = \min(1, 1 - t_A + t_B) = S_L(1 - t_A, t_B)$, where S_L is the Łukasiewicz t-conorm. Its minimal refinement function is therefore obtained from that of S_L applied to $(1 - t_A, t_B)$, with the change to the negated antecedent translated back to t_A:

$$\rho^*_{R_L}(t_A, t_B, 1) = \begin{bmatrix} t_A - \frac{1}{2}\max(t_A - t_B, 0) \\ t_B + \frac{1}{2}\max(t_A - t_B, 0) \end{bmatrix} \tag{B10}$$

lifts the superclass up and pushes the subclass down so that they meet in the middle.

$$\rho^*_{T_L}(t_A, t_B, 0) = \begin{bmatrix} t_A - \frac{1}{2}\max(t_A + t_B - 1, 0) \\ t_B - \frac{1}{2}\max(t_A + t_B - 1, 0) \end{bmatrix} \tag{B11}$$

pushes both disjoint classes down equally.

Appendix C Consistency of 3-step Inconsistency Resolution

In this section, a proof sketch for Proposition 1 is provided, showing that the 3-step inconsistency resolution algorithm leads to consistent predictions with regard to subsumption and disjointness axioms.

Given are input predictions $\mathbf{p}^e(x) \in \mathbb{R}^C$, a reflexive and transitive subsumption relation $\sqsubseteq$ and a set of disjoint class pairs $\{(a, b) | (a, b) \in C^2 \wedge a \cap b = \bot\}$. Regarding subsumption violations, only step 3 of Algorithm 1 is relevant (the purpose of step 1 is to take the predictions into account where possible). Here, for any class a that has a superclass b, the set of superclasses of a will be a superset of b's superclass set (due to the transitive nature of $\sqsubseteq$). Therefore, the minimum of predictions across a's superclass set will be equal to or lower than b's minimum. In other words, $p_a^{r3}(x) \leq p_b^{r3}(x)$ is guaranteed. Thus, if the prediction of a is positive, the prediction of b must be positive as well.

Disjointness relations are resolved by step 2. For each pair (a, b) of disjoint classes, it ensures that either $p_a^{r2}(x) \leq \theta^e$ or $p_b^{r2}(x) \leq \theta^e$. This still holds if the predictions are later changed (either by other disjoint pairs involving a or b or by step 3) since all changes are downward corrections. Therefore, $p_a^{r3}(x) \leq p_a^{r2}(x) \leq \theta^e$ or $p_b^{r3}(x) \leq p_b^{r2}(x) \leq \theta^e$. The same holds for class pairs that are subclasses of classes with an

Table D1: Comparison of Lopster and ChEBI for each class directly predicted by a Lopster rule.

ChEBI Class	Total	Lopster		ChEBI		Neither	Cohen's
		abs.	rel.	abs.	rel.		Kappa
37138 (astatine molecular entity)	0	0	-	0	-	18,968	1.00
22928 (bromine molecular entity)	210	210	1.00	51	0.24	18,758	0.39
23117 (chlorine molecular entity)	1,055	1,055	1.00	285	0.27	17,913	0.41
24062 (fluorine molecular entity)	1,073	1,073	1.00	163	0.15	17,895	0.25
24860 (iodine molecular entity)	56	56	1.00	20	0.36	18,912	0.53
24471 (halogen molecular entity)	2,259	2,258	1.00	538	0.24	16,709	0.35
36917 (polonium molecular entity)	4	4	1.00	0	0.00	18,964	0.00
26628 (selenium molecular entity)	20	20	1.00	7	0.35	18,948	0.52
26835 (sulfur molecular entity)	3,130	3,101	0.99	1,051	0.34	15,838	0.45
33305 (tellurium molecular entity)	7	7	1.00	6	0.86	18,961	0.92
33304 (chalcogen molecular entity)	18,335	18,333	1.00	13,300	0.73	633	0.15
36919 (antimony molecular entity)	2	2	1.00	0	0.00	18,966	0.00
22632 (arsenic molecular entity)	17	17	1.00	10	0.59	18,951	0.74
37196 (bismuth molecular entity)	3	3	1.00	2	0.67	18,965	0.80
51143 (nitrogen molecular entity)	11,915	11,915	1.00	7,631	0.64	7,053	0.57
33302 (pnictogen molecular entity)	12,367	12,365	1.00	8,489	0.69	6,601	0.60
23237 (chromium molecular entity)	5	5	1.00	5	1.00	18,963	1.00
25370 (molybdenum molecular entity)	8	8	1.00	7	0.88	18,960	0.93
37224 (seaborgium molecular entity)	0	0	-	0	-	18,968	1.00
33742 (tungsten molecular entity)	2	2	1.00	2	1.00	18,966	1.00
33741 (chromium group molecular entity)	15	15	1.00	14	0.93	18,953	0.97
33583 (noble gas molecular entity)	4	4	1.00	3	0.75	18,964	0.86
50860 (organic molecular entity)	18,771	18,752	1.00	18,082	0.96	197	0.35
25806 (oxygen molecular entity)	18,166	18,165	1.00	12,709	0.70	802	0.16
33608 (hydrogen molecular entity)	18,796	18,796	1.00	2,846	0.15	172	0.00
26082 (phosphorus molecular entity)	1,641	1,639	1.00	1,350	0.82	17,327	0.89
24835 (inorganic molecular entity)	218	216	0.99	87	0.40	18,750	0.56
24632 (hydrocarbon)	201	156	0.78	156	0.78	18,767	0.71
24472 (halohydrocarbon)	30	28	0.93	16	0.53	18,938	0.64
33238 (monoatomic entity)	47	47	1.00	19	0.40	18,921	0.58
33575 (carboxylic acid)	4,018	3,765	0.94	1,303	0.32	14,950	0.38
33308 (carboxylic ester)	4,002	3,902	0.98	1,565	0.39	14,966	0.48
17087 (ketone)	3,212	3,200	1.00	872	0.27	15,756	0.38
25710 (organophosphorus compound)	1,612	1,607	1.00	977	0.61	17,356	0.74
194538 (livermorium molecular entity)	1	1	1.00	0	0.00	18,967	0.00
194536 (moscovium molecular entity)	0	0	-	0	-	18,968	1.00

asserted disjointness relation: If $a' \sqsubseteq a$ and $b' \sqsubseteq b$, step 3 ensures that $p^{r3}_{a'}(x) \leq p^{r3}_{a}(x)$ and $p^{r3}_{b'}(x) \leq p^{r3}_{b}(x)$. From this, $p^{r3}_{a'}(x) \leq \theta^e$ or $p^{r3}_{b'}(x) \leq \theta^e$ follows trivially.

Appendix D Lopster Evaluation

Table D1 shows the evaluation of Lopster on the ChEBI test set. Since we know that ChEBI annotations are incomplete, we do not use ChEBI as a gold standard to compute an F1 score for Lopster. Rather, we focus on inter-rater agreement. For each

class, we compare the number of molecules belonging to the class according to the Lopster rules ("Lopster abs.") to the number of *is a* relations in ChEBI ("ChEBI abs."). "Total" shows the number of molecules assigned to the class by either method. The relative scores give the proportion of "Total" assigned by Lopster / ChEBI. Molecules that were not assigned to the class are counted as "Neither". Cohen's Kappa [69] is the inter-rater agreement between Lopster and ChEBI, with a higher score signalling closer agreement. It is calculated as

$$\kappa = \frac{2 \times (\text{Both} \times \text{Neither} + (\text{Both} - \text{ChEBI}) \times (\text{Lopster} - \text{Both}))}{\text{Lopster} \times (\text{Lopster} - \text{Both} + \text{Neither}) + \text{ChEBI} \times (\text{ChEBI} - \text{Both} + \text{Neither})} \tag{D12}$$

where

$$\text{Both} = \text{Lopster} + \text{ChEBI} - \text{Total}. \tag{D13}$$

For a derivation of this formula, see Appendix B of [8].

Appendix E Regex Patterns for ChEBI Class Grouping

In Section 5.4, feature-based label subsets have been extracted by matching regular expressions against ChEBI class names. Table E2 shows the regex patterns used for each group. These patterns have been generated with LLM assistance and have been reviewed by the authors.

Appendix F Class-wise F1 Scores

Table F3 shows the 15 classes with the largest gains / losses of the ensemble against the best DL base learner and a support of at least 50 (which equates to approximately 10 test positives). Some of the groups discussed in Section 5.4 have representatives here. For instance, sugars appear prominently as the largest gains (*TDP-sugar*, *dTDP-sugar*, *UDP-sugar*), but also in the losses (*mannooligosaccharide*).

References

[1] Degtyarenko, K., De Matos, P., Ennis, M., Hastings, J., Zbinden, M., McNaught, A., Alcántara, R., Darsow, M., Guedj, M., Ashburner, M.: ChEBI: a database and ontology for chemical entities of biological interest. Nucleic Acids Research **36**(Database issue), 344–350 (2008)

[2] Hastings, J., Owen, G., Dekker, A., Ennis, M., Kale, N., Muthukrishnan, V., Turner, S., Swainston, N., Mendes, P., Steinbeck, C.: ChEBI in 2016: Improved services and an expanding collection of metabolites. Nucleic Acids Research **44**(D1), 1214–1219 (2016)

[3] Kim, S., Chen, J., Cheng, T., Gindulyte, A., He, J., He, S., Li, Q., Shoemaker, B.A., Thiessen, P.A., Yu, B., Zaslavsky, L., Zhang, J., Bolton, E.E.: PubChem 2025 update. Nucleic Acids Research **53**(D1), 1516–1525 (2025)

Table E2: Regular expressions defining each label subset. A class belongs to a subset if a pattern matches its ChEBI name (case-insensitive). In some cases, a class can belong to multiple subsets.

Family	Case-insensitive regex, matched against the ChEBI class name
metal/element entity	`molecular entity|coordination entity|group molecular|organometal|metal cation|monocation|monoatomic|\bmetal\b|organotin|organolithium|organochromium|organotellurium|organoselen|organoarsen|\bselenium\b|\barsenic\b`
sugar/glycoside	`glycos|glucos|galact|mannos|sacchar|\bsugar\b|hexos|pentos|triose|fucos|sialo|glucuron|deoxy\w*ose|amino\w* sugar|glycan|furanose|pyranose`
steroid	`steroid|cholan|corticoster|glucocortic|\bestr|cholane|sterol|gibberell`
amino-acid derivative	`amino acid|alanine|cysteine|serine|lysine|proline|arginine|glutamine|glutamate|aspart|leucine|valine|glycine|threonine|tyrosine|amino-acid|L-alpha-amino`
flavonoid	`flavan|flavon|flavone|isoflav|anthocyan|homoflavon`
CoA/acyl-CoA	`-CoA|acyl-CoA|enoyl`
ceramide/sphingolipid	`ceramide|sphingo|glycosphingo`
peptide	`peptide|depsipep`
glycerophospholipid/acylglycerol	`glycero|phosphatidyl|acylglycerol|diacyl|monoacyl|diglyceride|triglyceride|\bCDP-|wax ester`
N-heterocycle	`azine|azole|azepine|pyrid|pyrrol|indol|quinol|imidazol|pteridin|purine|pyrimidin|oxazol|thiazol|triazol|indolizin|phthalazin|carbapenem|cephem`
fatty acid / acyl anion	`fatty acid|carboxylic acid|carboxylate|dicarboxyl|icosanoid|docosanoid|thromboxane|oxoacyl|\bfatty acyl`
S/Se-containing	`\bthio|sulf|thia|thie|\bselen`
terpenoid	`terpen|terpenoid|prenol|iridoid`
alkaloid	`alkaloid`

[4] Djoumbou Feunang, Y., Eisner, R., Knox, C., Chepelev, L., Hastings, J., Owen, G., Fahy, E., Steinbeck, C., Subramanian, S., Bolton, E., Greiner, R., Wishart, D.S.: ClassyFire: Automated chemical classification with a comprehensive, computable taxonomy. Journal of Cheminformatics **8**(1), 61 (2016) https://doi.org/10.1186/s13321-016-0174-y . Accessed 2025-07-18

[5] Flügel, S., Glauer, M., Neuhaus, F., Hastings, J.: When one logic is not enough: Integrating first-order annotations in OWL ontologies. Semantic Web **16**(2), 1–15 (2025)

[6] Kutz, O., Hastings, J., Mossakowski, T.: Modelling highly symmetrical molecules: Linking ontologies and graphs. In: International Conference on Artificial Intelligence: Methodology, Systems, and Applications, pp. 103–111 (2012). Springer

[7] Flügel, S., Glauer, M., Mossakowski, T., Neuhaus, F.: ChemLog: Making MSOL viable for ontological classification and learning. In: 5th International Joint Conference on Learning and Reasoning (accepted) (2025). https://doi.org/10.48550/

Table F3: ChEBI classes in $\mathbf{ChEBI}^{1,766}_{3-\mathbf{STAR}}$ where the ensemble (DES with ILR-Łukasiewicz) had the largest gains / losses (in terms of F1 score) compared to GAT-aug (3-STAR). Classes with fewer than 50 members have been excluded, as their results are less significant.

ChEBI class	Support	base F1	ens. F1	Δ F1 (pp)
largest gains (support ≥ 50)				
TDP-sugar (CHEBI:22080)	69	0.00	100.00	+100.00
dTDP-sugar (CHEBI:23557)	65	0.00	93.33	+93.33
UDP-sugar (CHEBI:17297)	80	0.00	85.71	+85.71
3-hydroxy fatty acid (CHEBI:59845)	60	20.00	87.50	+67.50
aminoglycoside antibiotic (CHEBI:22507)	61	0.00	66.67	+66.67
pyrimidine nucleotide-sugar (CHEBI:61109)	176	37.04	92.68	+55.65
nucleotide-sugar oxoanion (CHEBI:59737)	227	42.86	97.78	+54.92
oxopurine (CHEBI:25810)	61	40.00	93.33	+53.33
sphinganine-based sphingolipid (CHEBI:82829)	76	15.38	64.00	+48.62
deoxyaldopentose phosphate (CHEBI:23634)	68	33.33	80.00	+46.67
fatty acyl-L-carnitine (CHEBI:176910)	59	40.00	85.71	+45.71
2-deoxyribose phosphate (CHEBI:19569)	65	36.36	80.00	+43.64
cyclic ketal (CHEBI:59779)	92	42.86	85.71	+42.86
methanesulfonates (CHEBI:48544)	53	50.00	90.91	+40.91
steroid aldehyde (CHEBI:131565)	59	60.00	100.00	+40.00
largest losses (support ≥ 50)				
homopolymer macromolecule (CHEBI:37997)	121	33.33	0.00	-33.33
anilide (CHEBI:13248)	54	80.00	50.00	-30.00
mannooligosaccharide (CHEBI:25174)	62	54.55	28.57	-25.97
selenium molecular entity (CHEBI:26628)	60	72.73	47.06	-25.67
thiophenes (CHEBI:26961)	97	92.31	66.67	-25.64
11,12-saturated fatty acyl-CoA (CHEBI:85348)	78	60.00	37.50	-22.50
organic oxide (CHEBI:25701)	52	75.00	53.33	-21.67
nickel group molecular entity (CHEBI:33747)	72	90.91	71.43	-19.48
pnictogen hydride (CHEBI:35881)	73	85.71	66.67	-19.05
butan-4-olide (CHEBI:22950)	88	85.71	66.67	-19.05
17beta-hydroxy steroid (CHEBI:35343)	108	92.31	75.00	-17.31
amidine (CHEBI:2634)	51	66.67	50.00	-16.67
monounsaturated fatty acyl-CoA(4-) (CHEBI:77331)	59	66.67	50.00	-16.67
2-hydroxy monocarboxylic acid (CHEBI:49302)	122	87.50	71.43	-16.07
aromatic alcohol (CHEBI:33854)	124	96.55	81.48	-15.07

arXiv.2507.13987 . http://arxiv.org/abs/2507.13987 Accessed 2025-07-21

[8] Flügel, S., Mossakowski, T., Neuhaus, F., Pfanenstiel, E., Glauer, M., Haak, E., Malik, A., O'Boyle, N.M.: Defining Peptides in ChEBI. Research Square (2026). https://doi.org/10.21203/rs.3.rs-7767033/v1

[9] Magka, D., Krötzsch, M., Horrocks, I.: A rule-based ontological framework for the classification of molecules. Journal of Biomedical Semantics **5**(1), 17 (2014) https://doi.org/10.1186/2041-1480-5-17

[10] Mungall, C.J., Malik, A., Korn, D.R., Reese, J.T., O'Boyle, N.M., Hastings, J.: Chemical classification program synthesis using generative artificial intelligence. Journal of Cheminformatics **17**(1), 152 (2025)

[11] Hastings, J., Glauer, M., Memariani, A., Neuhaus, F., Mossakowski, T.: Learning chemistry: Exploring the suitability of machine learning for the task of structure-based chemical ontology classification. Journal of Cheminformatics **13**, 1–20 (2021)

[12] Glauer, M., Neuhaus, F., Flügel, S., Wosny, M., Mossakowski, T., Memariani, A., Schwerdt, J., Hastings, J.: Chebifier: Automating semantic classification in ChEBI to accelerate data-driven discovery. Digital Discovery **3**(5), 896–907 (2024)

[13] Glauer, M., Memariani, A., Neuhaus, F., Mossakowski, T., Hastings, J.: Interpretable ontology extension in chemistry. Semantic Web **15**(4), 937–958 (2024)

[14] Memariani, A., Glauer, M., Flügel, S., Neuhaus, F., Hastings, J., Mossakowski, T.: Box embeddings for extending ontologies: a data-driven and interpretable approach. Journal of Cheminformatics **17**(1), 138 (2025)

[15] Flügel, S., Glauer, M., Hastings, J., Mossakowski, T., Mungall, C.J., Tumescheit, C., Neuhaus, F.: Chebifier 2: An Ensemble for Chemistry. In: SymGenAI4Sci 2025: First International Workshop on Symbolic and Generative AI for Science, Co-located with SEMANTiCS 2025 (2025)

[16] Deng, J., Ding, N., Jia, Y., Frome, A., Murphy, K., Bengio, S., Li, Y., Neven, H., Adam, H.: Large-scale object classification using label relation graphs. In: European Conference on Computer Vision (ECCV), pp. 48–64 (2014). Springer

[17] Daniele, A., Van Krieken, E., Serafini, L., Van Harmelen, F.: Refining neural network predictions using background knowledge. Machine Learning **112**(9), 3293–3331 (2023)

[18] Flügel, S., Khedekar, A.G.: Chebifier: An AI ensemble model for predicting chemical classes in the ChEBI ontology. GitHub repository. Version 1.3.0. Accessed 2026-10-05 (2026). https://github.com/ChEB-AI/python-chebifier

[19] Glauer, M., Flügel, S., Khedekar, A.G., Hastings, J.: Chebifier 2 web interface. Accessed 2026-10-05 (2026). https://chebifier.hastingslab.org/

[20] Moyano, J.M., Gibaja, E.L., Cios, K.J., Ventura, S.: Review of ensembles of multi-label classifiers: Models, experimental study and prospects. Information Fusion **44**, 33–45 (2018) https://doi.org/10.1016/j.inffus.2017.12.001

[21] Read, J., Pfahringer, B., Holmes, G., Frank, E.: Classifier chains for multi-label classification. Machine Learning **85**(3), 333–359 (2011)

[22] Tsoumakas, G., Katakis, I., Vlahavas, I.: Effective and efficient multilabel classification in domains with large number of labels. In: Proc. ECML/PKDD 2008 Workshop on Mining Multidimensional Data (MMD'08), vol. 21, pp. 53–59 (2008)

[23] Schapire, R.E., Singer, Y.: BoosTexter: A boosting-based system for text categorization. Machine Learning **39**(2), 135–168 (2000)

[24] Kocev, D., Vens, C., Struyf, J., Džeroski, S.: Ensembles of multi-objective decision trees. In: European Conference on Machine Learning (ECML), pp. 624–631 (2007). Springer

[25] Blockeel, H., De Raedt, L., Ramon, J.: Top-down induction of clustering trees. In: Shavlik, J.W. (ed.) Proceedings of the 15th International Conference on Machine Learning (ICML), pp. 55–63. Morgan Kaufmann, San Francisco, CA (1998)

[26] Tahir, M.A., Kittler, J., Bouridane, A.: Multilabel classification using heterogeneous ensemble of multi-label classifiers. Pattern Recognition Letters **33**(5), 513–523 (2012) https://doi.org/10.1016/j.patrec.2011.10.019

[27] Dogan, A., Birant, D.: A weighted majority voting ensemble approach for classification. In: 2019 4th International Conference on Computer Science and Engineering (UBMK), pp. 1–6 (2019). https://doi.org/10.1109/UBMK.2019.8907028 . IEEE. https://ieeexplore.ieee.org/abstract/document/8907028

[28] Sterniczuk, B., Charytanowicz, M.: An Ensemble Transfer Learning Model for Brain Tumors Classification using Convolutional Neural Networks. Advances in Science and Technology Research Journal **18**(8), 204–216 (2024) https://doi.org/10.12913/22998624/193627

[29] Dinca, M.A., Popescu, D., Ichim, L., Angelescu, N.: Ensemble of efficient vision transformers for insect classification. Applied Sciences **15**(13) (2025)

[30] You, R., Zhang, Z., Xiong, Y., Sun, F., Mamitsuka, H., Zhu, S.: GOLabeler: improving sequence-based large-scale protein function prediction by learning to rank. Bioinformatics **34**(14), 2465–2473 (2018)

[31] Ashburner, M., Ball, C.A., Blake, J.A., Botstein, D., Butler, H., Cherry, J.M., Davis, A.P., Dolinski, K., Dwight, S.S., Eppig, J.T., *et al.*: Gene Ontology: tool for the unification of biology. Nature Genetics **25**(1), 25–29 (2000)

[32] Aleksander, S.A., Balhoff, J.P., Carbon, S., Cherry, J.M., Ebert, D., Feuermann, M., Gaudet, P., Harris, N.L., Hill, D.P., Kalita, P., *et al.*: The Gene Ontology knowledgebase in 2026. Nucleic Acids Research **54**(D1), 1292 (2026)

[33] Li, H.: A short introduction to learning to rank. IEICE Transactions on Information and Systems **E94-D**(10), 1854–1862 (2011)

[34] Yuan, Q., Gao, J., Wu, D., Zhang, S., Mamitsuka, H., Zhu, S.: DrugE-Rank: improving drug–target interaction prediction of new candidate drugs or targets by ensemble learning to rank. Bioinformatics **32**(12), 18–27 (2016)

[35] Ko, A.H.R., Sabourin, R., Britto Jr., A.S.: From dynamic classifier selection to dynamic ensemble selection. Pattern Recognition **41**(5), 1718–1731 (2008)

[36] Cruz, R.M.O., Sabourin, R., Cavalcanti, G.D.C.: Dynamic classifier selection: Recent advances and perspectives. Information Fusion **41**, 195–216 (2018)

[37] Cruz, R.M.O., Sabourin, R., Cavalcanti, G.D.C., Ren, T.I.: META-DES: A dynamic ensemble selection framework using meta-learning. Pattern Recognition **48**(5), 1925–1935 (2015)

[38] Cruz, R.M.O., Sabourin, R., Cavalcanti, G.D.C.: META-DES.H: A dynamic ensemble selection technique using meta-learning and a dynamic weighting approach. In: 2015 International Joint Conference on Neural Networks (IJCNN), pp. 1–8 (2015). IEEE

[39] Obozinski, G., Lanckriet, G., Grant, C., Jordan, M.I., Noble, W.S.: Consistent probabilistic outputs for protein function prediction. Genome Biology **9**(Suppl 1), 6 (2008)

[40] Giunchiglia, E., Lukasiewicz, T.: Coherent hierarchical multi-label classification networks. Advances in Neural Information Processing Systems **33**, 9662–9673 (2020)

[41] Kourmpetis, Y.A.I., van Dijk, A.D.J., ter Braak, C.J.F.: Gene Ontology consistent protein function prediction: the FALCON algorithm applied to six eukaryotic genomes. Algorithms for Molecular Biology **8**(1), 10 (2013)

[42] Jiang, R., Liu, M., Liu, H., Yu, Y.: Hierarchy-consistent learning and adaptive loss balancing for hierarchical multi-label classification. In: Proceedings of the 34th ACM International Conference on Information and Knowledge Management, pp. 1190–1199 (2025)

[43] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., Fei-Fei, L.: ImageNet: A large-scale hierarchical image database. In: 2009 IEEE Conference on Computer Vision and Pattern Recognition, pp. 248–255 (2009). IEEE

[44] Mienye, I.D., Sun, Y.: A Survey of Ensemble Learning: Concepts, Algorithms, Applications, and Prospects. IEEE Access **10**, 99129–99149 (2022) https://doi.org/10.1109/ACCESS.2022.3207287

[45] Littlestone, N., Warmuth, M.K.: The Weighted Majority Algorithm. Information and Computation **108**(2), 212–261 (1994) https://doi.org/10.1006/inco.1994.1009

[46] Morgan, H.L.: The generation of a unique machine description for chemical structures-a technique developed at chemical abstracts service. Journal of chemical documentation **5**(2), 107–113 (1965)

[47] Bajusz, D., Rácz, A., Héberger, K.: Why is tanimoto index an appropriate choice for fingerprint-based similarity calculations? Journal of Cheminformatics **7**(1), 20 (2015)

[48] Burges, C.J.C.: From RankNet to LambdaRank to LambdaMART: An overview. Technical Report MSR-TR-2010-82, Microsoft Research (2010)

[49] Platt, J.C.: Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. In: Smola, A.J., Bartlett, P., Schölkopf, B., Schuurmans, D. (eds.) Advances in Large Margin Classifiers, pp. 61–74. MIT Press, Cambridge, MA (1999)

[50] Khedekar, A.G.: Integrating Chemical Knowledge into Graph Neural Networks. Master's thesis, University of Osnabrück (2025). https://www.uni-osnabrueck.de/fileadmin/informatik/Arbeitsgruppen/Hybride_KI/mt_aditya_khedekar.pdf

[51] EMBL-EBI: ChEBI disjointness axioms module. Accessed 2026-10-05. https://ftp.ebi.ac.uk/pub/databases/chebi/ontology/chebi-disjoints.owl

[52] Flügel, Simon: Additional ChEBI disjointness axioms. Accessed 2026-10-05. https://huggingface.co/datasets/chebai/chebifier/blob/main/disjoint_additional.csv

[53] Rong, Y., Bian, Y., Xu, T., Xie, W., Wei, Y., Huang, W., Huang, J.: Self-supervised graph transformer on large-scale molecular data. In: Advances in Neural Information Processing Systems, vol. 33, pp. 12559–12571. Curran Associates, Inc., Red Hook, NY (2020). https://proceedings.neurips.cc/paper/2020/hash/94aef38441efa3380a3bed3faf1f9d5d-Abstract.html Accessed 2026-02-03

[54] Wang, Y., Li, Z., Barati Farimani, A.: Graph Neural Networks for Molecules. In: Qu, C., Liu, H. (eds.) Machine Learning in Molecular Sciences, pp. 21–66. Springer, Cham (2023). https://doi.org/10.1007/978-3-031-37196-7_2

[55] Khedekar, A.G., Flügel, S., Glauer, M.: ChEB-AI Graph: Extension of ChEB-AI for Graph Neural Networks. GitHub repository. Version 1.2.3. Accessed 2026-10-05 (2026). https://github.com/ChEB-AI/python-chebai-graph

[56] Bresson, X., Laurent, T.: Residual Gated Graph ConvNets. arXiv. arXiv:1711.07553 [cs] (2018). https://doi.org/10.48550/arXiv.1711.07553 . http://arxiv.org/abs/1711.07553 Accessed 2025-07-18

[57] Scarselli, F., Gori, M., Tsoi, A.C., Hagenbuchner, M., Monfardini, G.: The graph neural network model. IEEE Transactions on Neural Networks **20**(1), 61–80

(2009)

[58] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Liò, P., Bengio, Y.: Graph attention networks. In: International Conference on Learning Representations (ICLR) (2018). https://doi.org/10.48550/arXiv.1710.10903 . http://arxiv.org/abs/1710.10903 Accessed 2026-02-10

[59] Brody, S., Alon, U., Yahav, E.: How attentive are graph attention networks? In: International Conference on Learning Representations (ICLR) (2022). https://arxiv.org/abs/2105.14491

[60] Flügel, S., Khedekar, A.G.: ChemLog: Rule-based classification of peptides in ChEBI. GitHub repository. Version 1.0.7. Accessed 2026-10-05 (2025). https://github.com/sfluegel05/chemlog-peptides

[61] Flügel, S., Khedekar, A.G.: ChemLog-extra: Extension of ChemLog to a broad range of ChEBI classes. GitHub repository. Version 1.1. Accessed 2026-10-05 (2026). https://github.com/ChEB-AI/chemlog-extra

[62] Hitzler, P., Krötzsch, M., Parsia, B., Patel-Schneider, P.F., Rudolph, S.: OWL 2 Web Ontology Language Primer. W3C Recommendation (2009). https://www.w3.org/TR/owl2-primer/

[63] Magka, D.: LoPStER: A Logic Programming for Structured Entites Reasoner. GitHub repository. Accessed 2026-10-05 (2014). https://github.com/magkades/lopster/

[64] Leone, N., Pfeifer, G., Faber, W., Eiter, T., Gottlob, G., Perri, S., Scarcello, F.: The DLV system for knowledge representation and reasoning. ACM Transactions on Computational Logic (TOCL) **7**(3), 499–562 (2006)

[65] Gebser, M., Kaminski, R., Kaufmann, B., Schaub, T.: Multi-shot ASP solving with clingo. Theory and Practice of Logic Programming **19**(1), 27–82 (2019)

[66] Ridnik, T., Ben-Baruch, E., Zamir, N., Noy, A., Friedman, I., Protter, M., Zelnik-Manor, L.: Asymmetric loss for multi-label classification. In: 2021 IEEE/CVF International Conference on Computer Vision (ICCV), pp. 82–91 (2021). IEEE

[67] Brinkmann, T.: Focal Loss for Molecule Classification. https://www.uni-osnabrueck.de/fileadmin/informatik/Arbeitsgruppen/Hybride_KI/bt_tim_brinkmann.pdf

[68] Flügel, S., Mossakowski, T.: HyDI: A hybrid Deep Learning-Inductive Logic Programming ensemble for multi-label classification (2026). https://arxiv.org/abs/2609.37740

[69] Cohen, J.: A coefficient of agreement for nominal scales. Educational and

Psychological Measurement **20**(1), 37–46 (1960)