KNOWLEDGE-DRIVEN ACTIVE LEARNING

Anonymous authors
Paper under double-blind review

ABSTRACT

The deployment of Deep Learning (DL) models is still precluded in those contexts where the amount of supervised data is limited. To answer this issue, active learning strategies aim at minimizing the amount of labelled data required to train a DL model. Most active strategies are based on uncertain sample selection, and even often restricted to samples lying close to the decision boundary. These techniques are theoretically sound, but an understanding of the selected samples based on their content is not straightforward, further driving non-experts to consider DL as a black-box. For the first time, here we propose a different approach, taking into consideration common domain-knowledge and enabling non-expert users to train a model with fewer samples. In our Knowledge-driven Active Learning (KAL) framework, rule-based knowledge is converted into logic constraints and their violation is checked as a natural guide for sample selection. We show that even simple relationships among data and output classes offer a way to spot predictions for which the model need supervision. The proposed approach (i) outperforms many active learning strategies in terms of average F1 score, particularly in those contexts where domain knowledge is rich. Furthermore, we empirically demonstrate that (ii) KAL discovers data distribution lying far from the initial training data unlike uncertainty-based strategies, (iii) it ensures domain experts that the provided knowledge is respected by the model on test data, and (iv) it can be employed even when domain-knowledge is not available by coupling it with a XAI technique. Finally, we also show that KAL is also suitable for object recognition tasks and, its computational demand is low, unlike many recent active learning strategies.

1 INTRODUCTION

Deep Learning (DL) methods have achieved impressive results over the past decade in fields ranging from computer vision to text generation (LeCun et al., 2015). However, most of these contributions relied on overly data-intensive models (e.g. Transformers), trained on huge amounts of data (Marcus, 2018). With the advent of Big Data, sample collection does not represent an issue any more, but, nonetheless, in some contexts the number of *supervised* data is limited, and manual labelling can be expensive (Yu et al., 2015). Therefore, a common situation is the unlabelled pool scenario (McCallumzy & Nigamy, 1998), where many data are available, but only some are annotated. Historically, two strategies have been devised to tackle this situation: semi-supervised learning which exploit the unlabelled data to enrich feature representations (Zhu & Goldberg, 2009), and active learning which select the smallest set of data to annotate to improve the most model performances (Settles, 2009).

The main assumption behind active learning strategies is that there exists a subset of samples that allows to train a model with a similar accuracy as when fed with all training data. Iteratively, the strategy indicates the optimal samples to be annotated from the unlabelled pool. This is generally done by ranking the unlabelled samples w.r.t. a given measure, usually on the model predictions (Settles, 2009; Netzer et al., 2011; Wang & Shang, 2014), or on the input data distribution (Zhdanov, 2019; Santoro et al., 2017) and by selecting the samples associated to the highest rankings (Ren et al., 2021; Zhan et al., 2021). While being theoretically sound, an understanding of the selected samples based on their content is not straightforward, in particular to non-ML experts. This issue becomes particularly relevant when considering that Deep Neural Networks are already seen as black box models (Gilpin et al., 2018; Das & Rad, 2020) On the contrary, we believe that neural models must be linked to Commonsense knowledge related to a given learning problem. Therefore, in this paper, we propose for the first time to exploit this symbolic knowledge in the selection process of an active

learning strategy. This not only lower the amount of supervised data, but it also enables domain experts to train a model leveraging their knowledge. More precisely, we propose to compare the predictions over the unsupervised data with the available knowledge and to exploit the inconsistencies as a criterion for selecting the data to be annotated. Domain knowledge, indeed, can be expressed as First-Order Logic (FOL) clauses and translated into real-valued logic constraints by means of T-Norms (Klement et al., 2013) to assess its satisfaction (Gnecco et al., 2015; Diligenti et al., 2017; Melacci et al., 2021).

In the experiments, we show that the proposed Knowledge-driven Active Learning (KAL) strategy (i) performs better (on average) than several standard active learning methods, particularly in those contexts where domain-knowledge is rich. We empirically demonstrate (ii) that this is mainly due to the fact that the proposed strategy allows discovering data distributions lying far from the initial training data, unlike uncertainty-based approaches. Furthermore, we show that (iii) the provided knowledge is acquired by the trained model and respected on unseen data, (iv) that KAL can also work on domains where no knowledge is available if we extract knowledge from the same model by means of a XAI technique, (v) that the KAL strategy can be easily employed also in the object-detection context, where standard uncertainty-based strategies are not-straightforward to apply (Haussmann et al., 2020) and, finally, (vi) that the proposed method is not computationally expensive, unlike many recent methods.

The paper is organized as follows: in Section 2 the proposed method is explained in details, with first an example on inferring the XOR operation and then contextualized in more realistic active learning domains; the aforementioned experimental results on different datasets are reported in Section 3, comparing the proposed technique with several active learning strategies; in Section 4 the related work about active learning and about integrating reasoning with machine learning is briefly resumed; finally, in Section 5 we conclude the paper by considering possible future work.

2 Knowledge-driven Active Learning

In this paper, we focus on classification problems with c classes, in which each input $x \in X$ is associated to one or more classes and d represents the input dimensionality. Let us consider the classification problem $f\colon X \to Y$, where $X \subseteq \mathbb{R}^d$ represents the feature space which may also comprehend non-structured data (e.g. input images) and $Y \subseteq \{0,1\}^c$ is the output space composed of $c \geq 1$ dimensions. More precisely, we consider a vector function $f = [f_1, \ldots, f_c]$, where each function f_i predicts the probability that x belongs to the i-th class. In the Active Learning context, we also define $X_s \subset X$ as the portion of input data already associated to an annotation $y_i \in Y_s \subset Y$ and n the dimensionality of the starting set of labelled data. At each iteration, a set of p samples $X_p \subset X_u \subset X$ is selected by the active learning strategy to be annotated from X_u, the unlabelled data pool, and be added to X_s. This process is repeated for q iterations, after which the training terminates. The maximum budget of annotations b therefore amounts to $b = n + q * p$. When considering an object-detection problem, for a given class i and a given image x_j, we consider as class membership probability, the maximum score value among all predicted bounding boxes around the objects belonging to the i-th class. Formally, $f_i(x_j) = \max_{s \in \mathcal{S}^i(x_j)} s(x_j)$ where $\mathcal{S}^i(x_j)$ is the set of the confidence scores of the bounding boxes predicting the i-th class for sample x_j.

Let us also consider the case in which additional *domain knowledge* is available for the problem at hand, involving relationships between data and classes. By considering the logic predicate $\mathbf{f}$ associated to each function f, First-Order Logic (FOL) becomes the natural way of describing these relationships. For example, $\forall x \in X,\ \mathbf{x_1}(x) \wedge \mathbf{x_2}(x) \Rightarrow \mathbf{f}(x)$, where $\mathbf{x_1}(x), \mathbf{x_2}(x)$ respectively represent the logic predicates associated to the first and the second input features, and meaning that when both predicates are true also the output function $f(x)$ needs to be true. Also, we can consider relations among classes, such as $\forall x \in X,\ \mathbf{f_v}(\mathbf{x}) \wedge \mathbf{f_z}(\mathbf{x}) \Rightarrow \mathbf{f_u}(\mathbf{x})$, meaning that the intersection between the v-th class and the z-th class is always included in the u-th one.

2.1 Converting Domain-Knowledge into loss functions

The Learning from Constraints framework (Gnecco et al., 2015; Diligenti et al., 2017) defines a way to convert domain knowledge into logic constraints and how to use them on the learning problem. Among a variety of other type of constraints (see, e.g., Table 2 in (Gnecco et al., 2015)), it studies the

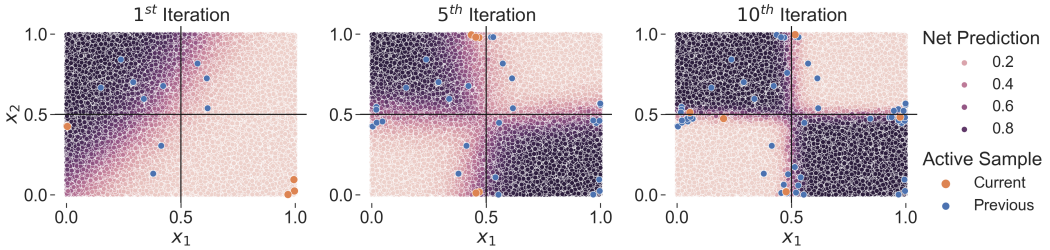

Figure 1: A visual example of KAL working principles on the *XOR-like* problem. We depict network predictions with different colour degrees (light colours negative predictions, dark colours positive prediction). Also, we depict in orange the samples selected in the current iteration and in blue those selected in previous iterations. From left, the scenario at the 1st, 5th and 10th iteration. Notice how the proposed method immediately discovers the unseen data distribution (right-bottom quadrant).

process of handling FOL formulas so that they can be either injected into the learning problem (in semi-supervised learning) or used as a knowledge verification measure (as in (Melacci et al., 2021) and in the proposed method). Going into more details, the FOL formulas representing the domain knowledge are converted into numerical constraints using the Triangular Norms (T-Norms, (Klement et al., 2013)). These binary functions generalize the conjunction operator $\wedge$ and offer a way to mathematically compute the satisfaction level of a given rule. Based on this assumption, we can detect whether the predictions made by the model on unlabelled data are coherent with the domain knowledge or not and possibly select these data to be annotated within an active learning strategy.

Following the previous example, $\mathbf{x_1}(x) \wedge \mathbf{x_2}(x) \Rightarrow \mathbf{f}(x)$[1] is converted into a bilateral constraint $\phi(f(x)) = 1$ that, by employing the product T-Norm, is $1 - \mathbf{x_1}(x)\mathbf{x_2}(x)(1 - \mathbf{f}(x)) = 1$. With $\varphi(f(x)) = 1 - \phi(f(x))$ we indicate the loss function associated to the bilateral constraints, which measures the level of satisfaction of the given constraints and has its minimum value in zero. Again, recalling the previous example, the associated loss function would be $\varphi(f(x)) = \mathbf{x_1}(x)\mathbf{x_2}(x)(1 - \mathbf{f}(x))$, which indeed is satisfied when either $\mathbf{x_1}(x)$ or $\mathbf{x_2}(x)$ is zero or $\mathbf{f}(x)$ is approximately one. Finally, the loss function considering all the available FOL formulas $\mathcal{K}$ for the given problem is computed to drive the active strategy, selecting the points $x^\star$ which violate the most the constraints.

$$KAL: \qquad x^\star = \arg\max_{x \in X_u} \sum_{k \in \mathcal{K}} \varphi_k(f(x)) \tag{1}$$

At each iteration, the KAL strategy selects p samples $x^\star$ to annotate from the unlabelled pool X_u.

2.2 An intuitive example: the *XOR-like* problem

A well-known problem in machine learning is the inference of the eXclusive OR (XOR) operation. To show the working principles of the proposed approach, we propose a variant of this experiment, in which a neural network learns the *XOR-like* operation from a distribution of non-boolean samples. Specifically, we sampled 10^5 points from the distribution $x \in [0, 1]^2$, and we assigned a label $y(x)$ as following:

$$y(x) = \begin{cases} 1, & \text{if } (x_1 > 0.5 \wedge x_2 \leq 0.5) \vee (x_1 \leq 0.5 \wedge x_2 > 0.5), \\ 0, & \text{otherwise} \end{cases} \tag{2}$$

Also, we express the XOR operation through a FOL formula $(\mathbf{x_1} \wedge \neg\mathbf{x_2}) \vee (\neg\mathbf{x_1} \wedge \mathbf{x_2}) \Leftrightarrow \mathbf{f}$. For the sake of simplicity, here and in the following, we dropped the argument (x) of the logic predicates. As seen before, through the T-Norm operation we can convert the logic rule into a numerical constraint, and we can calculate its violation through the loss functions:

$$\begin{aligned} \varphi_{\mathbf{x_1} \oplus \mathbf{x_2} \rightarrow \mathbf{f}} &= (\mathbf{x_1} + \mathbf{x_2} - 2\mathbf{x_1}\mathbf{x_2})(1 - \mathbf{f}), \\ \varphi_{\mathbf{f} \rightarrow \mathbf{x_1} \oplus \mathbf{x_1}} &= \mathbf{f}(1 - (\mathbf{x_1} + \mathbf{x_2} - 2\mathbf{x_1}\mathbf{x_2})) \end{aligned} \tag{3}$$

each one representing the loss associated to one direction of the double implication. For an automatic computation of the loss functions associated to rule violations, please refer to (Marra et al., 2019).

[1] Practically, the predicate $\mathbf{x_i}$ is obtained by applying a steep sigmoid function over the i-th input feature x_i.

In Fig. 1, we reported an example of the proposed strategy starting from $n = 10$ labelled data and by selecting $p = 5$ samples at each iteration violating the most Eq. 3, and for $q = 100$ iterations. We depict network predictions with different colour degrees, from light beige (negative predictions) to black (positive predictions); we depict in orange the points selected at current iterations and in blue those selected previously. Notice how the proposed method immediately discovers novel data distributions by selecting samples from the right-bottom quadrant (orange points—figure on the left). After 10 iterations (figure at the centre) the network has mostly learnt the correct data distribution. Later, the proposed strategy refines network predictions by sampling along the decision boundaries (blue points—figure on the right), allowing the network to almost already solve the learning problem (accuracy $\sim 100\%$) in just 10 iterations. As will be seen in the next section, instead, standard random selection (but also uncertainty-based ones) will require many more iterations.

2.3 Real-life scenario: partial knowledge and different type of rules

It is clear that, in the case of the *XOR-like* problem, the knowledge is complete: if we compute the predictions directly through the rule, we already solve the learning problem. However, the purpose of that simple experiment is to show the potentiality of the proposed approach in integrating the available symbolic knowledge into a learning problem. In real-life scenarios, such a situation is unrealistic, but still we might have access to some partial useful knowledge that may allow solving more quickly a given learning problem. Also, it may promote an easier interaction of domain experts with DL model by asking them to provide their knowledge rather than just data annotations.

More precisely, when we consider structured data (e.g., tabular data), a domain expert may know some very simple relations taking into consideration few features and the output class. This knowledge may not be sufficient to solve the learning problem, but a KAL strategy can still exploit it to drive the network to a fast convergence, as we will see in Section 3. On the opposite, when we consider unstructured data (e.g., images or audio signals) the employed knowledge cannot directly rely on the input features. Nonetheless, when considering a multi-label learning problem, a user may know in advance some relations between the output classes. Let us consider, as an example, a Dog-vs-Person classification: we might know that a main object (e.g., a dog) is composed of several parts (e.g., a muzzle, a body, a tail). A straightforward translation of this compositional property into a FOL rule is $\mathbf{Dog} \Rightarrow \mathbf{Muzzle} \vee \mathbf{Body} \vee \mathbf{Tail}$. Formulating the composition in the opposite way is correct as well (e.g., $\mathbf{Muzzle} \Rightarrow \mathbf{Dog}$). Also, in all classification problems, at least one of the main classes needs to be predicted (e.g., $\mathbf{Dog} \vee \mathbf{Person}$). Finally, we can always incorporate an uncertainty-like rule requiring each predicate to be either true or false, i.e., $\mathbf{Dog} \oplus \neg\mathbf{Dog}$.

3 Experiments

In this work, we considered three different learning scenarios, comparing the proposed technique with several standard active strategies. We evaluated the proposed method on two standard problems from machine learning textbooks (Bishop, 2006), the inference of the *XOR-like* problem and the classification of *IRIS* plants given their characteristics. We also evaluated our method on two standard image-classification tasks: the *ANIMALS* dataset, extracted from ImageNet (Deng et al., 2009), and the Caltech-UCSD Birds-200-2011 dataset (*CUB200*, (Wah et al., 2011)). As a proof of concept, we also analysed the performances of the KAL when employed in the simple *DOGvsPERSON* object recognition task. For more details regarding the latter, please refer to Appendix A.1. For each dataset, n, p, q, as well as the number of training epochs and the network structure are fixed in advance according to the number of classes, the dataset size and the task complexity. Reported average results are computed on the test sets of a k-fold Cross Validation (with $k = 10$ in the tabular data problems and $k = 5$ in the computer vision ones). More details regarding each experimental problem, as well as a table reporting all the employed rules, are available in Appendix A.2. The code will be published on a publicly available repository, and it is provided in the supplementary material. A simple code example is also reported in Appendix B showing how to solve the *XOR-like* problem with the KAL strategy. All experiments were run on an Intel i7-9750H CPU machine with an NVIDIA 2080 RTX GPU and 64 GB of RAM.

Table 1: Comparison of the methods in terms of the mean F1 score AUBC and standard deviation when increasing the number of labelled points. On top, starting and ending labelling budget for each dataset. The three best results are reported in bold. NA indicates strategies too expensive to compute.

	Dataset	XOR	IRIS	Animals	CUB
Strategy	Budget	10-400	10-70	100-2500	2000-7000
ADV_{BIM}		89.79 $\pm$12.14	78.81 $\pm$21.02	53.93 $\pm$0.42	50.82 $\pm$0.36
$ADV_{DEEPFOOL}$		87.56 $\pm$13.08	78.86 $\pm$21.04	NA	NA
BALD		83.94 $\pm$9.25	90.96 $\pm$6.83	53.74 $\pm$1.96	51.17 $\pm$0.62
KCENTER		**96.79** $\pm$0.43	**92.65** $\pm$3.99	43.37 $\pm$3.29	48.86 $\pm$0.52
KMEANS		83.73 $\pm$4.40	88.55 $\pm$8.10	52.87 $\pm$1.18	49.96 $\pm$0.43
Entropy		90.11 $\pm$9.61	82.11 $\pm$18.49	**57.26** $\pm$0.94	51.91 $\pm$0.12
$Entropy_D$		87.80 $\pm$11.09	81.91 $\pm$18.36	56.56 $\pm$0.95	**51.92** $\pm$0.39
LeastConf		89.97 $\pm$10.55	84.35 $\pm$13.89	53.33 $\pm$2.12	50.20 $\pm$0.43
$LeastConf_D$		89.98 $\pm$12.05	84.83 $\pm$14.03	53.85 $\pm$0.84	50.07 $\pm$0.45
Margin		89.97 $\pm$10.55	85.46 $\pm$14.88	**57.26** $\pm$1.66	51.69 $\pm$0.41
$Margin_D$		89.98 $\pm$12.05	84.94 $\pm$14.32	**57.03** $\pm$0.42	51.54 $\pm$0.48
Random		96.18 $\pm$0.55	92.04 $\pm$5.14	52.62 $\pm$0.49	50.29 $\pm$0.38
SupLoss		96.77 $\pm$0.66	92.48 $\pm$3.34	54.97 $\pm$2.10	49.53 $\pm$0.35
KAL		**97.73** $\pm$0.83	**93.60** $\pm$3.87	56.15 $\pm$1.04	**51.98** $\pm$0.35
KAL_D		**97.96** $\pm$0.79	**93.32** $\pm$4.13	55.56 $\pm$1.21	**52.10** $\pm$0.24

Compared methods We compared KAL with 12 active learning strategies commonly considered in literature (Ren et al., 2021; Zhan et al., 2022). As representatives of uncertainty-based strategies, we considered **Entropy** (Settles, 2009) selecting samples associated to predictions having maximum entropy, **Margin** (Netzer et al., 2011) predictions with minimum margin between the top-two classes, and **LeastConf** (Wang & Shang, 2014) predictions with the lowest confidences, together with their Monte Carlo Dropout versions (Beluch et al., 2018) (respectively $\mathbf{Entropy}_D$, $\mathbf{Margin}_D$, $\mathbf{LeastConf}_D$). As more recent uncertainty-based strategies, we compared with Bayesian Active Learning by Disagreements **BALD** (Gal et al., 2017), with two strategies computing the margin by means of adversarial attacks $\mathbf{ADV}_{DEEPFOOL}$ (Ducoffe & Precioso, 2018), $\mathbf{ADV}_{BIM}$ (Zhan et al., 2022) and with **SupLoss** a simplified upper bound of the method proposed in (Yoo & Kweon, 2019) employing the actual labels (available only on benchmarks). As Diversity-based methods, we selected **KMeans** (Zhdanov, 2019) and **KCenter** a greedy version of the CoreSet method (Sener & Savarese, 2017). More details regarding each method are reported in Appendix A.3, together with a table resuming the associated losses. For the method implementations, we mostly relied on the DeepAL library (Huang, 2021).

3.1 KAL PROVIDES BETTER PERFORMANCE ON AVERAGE THAN ANY OTHER STRATEGY.

For a quantitative comparison of the different methods, we evaluated the performances of the different active learning strategies. In Figure 2 we reported the average F1 scores budget curves when increasing the number of selected labelled data. In Table 1 we also report the Area Under the Budget Curves (AUBC), as defined in Zhan et al. (2021). In both the *XOR-like* and *IRIS* learning problems, we can observe how both the proposed methods KAL and KAL_D (the corresponding Monte-Carlo Dropout version) have the highest average AUC with a 7-12 % increment over standard uncertainty-based strategies in both cases. The only competitive method in both cases is the CoreSet-based approach KCenter. This behaviour will be better analysed in Section 3.3. Interestingly, when analysing the top plots in Fig. 2 we can appreciate how the proposed methods enable the network to learn more quickly the given tasks w.r.t. the other ones. While this was an expected behaviour on the *XOR-like* task since the provided rules completely explain the learning problem, on the *IRIS* classification task it is surprising since only 3 simple rules considering a maximum of 2 features each are given (e.g., $\neg\mathbf{Long_Petal} \Rightarrow \mathbf{Setosa}$). A slightly-different situation can be observed in the image classification tasks (bottom plots in Fig. 2). Here we notice the importance of employing well-structured knowledge. In the *ANIMALS* task, indeed, only 17 rules are employed relating animal species and their characteristics (e.g., $\mathbf{Fly} \Rightarrow \neg\mathbf{Penguin}$). In this case, the results with KAL are

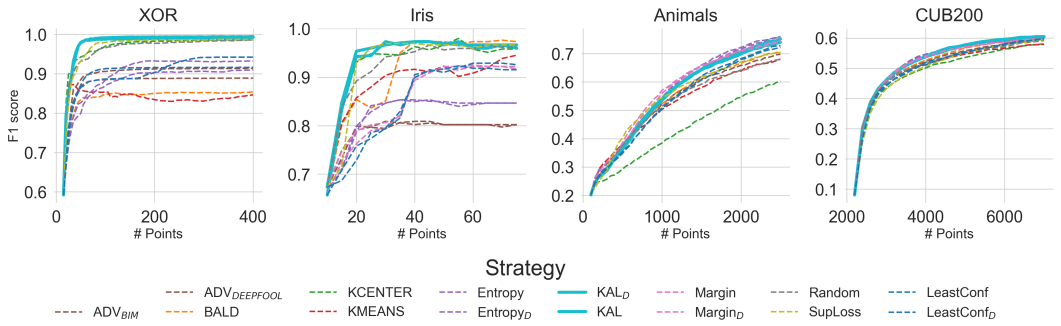

Figure 2: Average test F1 score performance growth on the four experiments when increasing the number of labelled samples. Confidence intervals have not been reported for better readability. Method variants (e.g., the Monte-Carlo Dropout versions) are displayed with the same colour.

only on average w.r.t. uncertainty-based approaches (better than LeastConf, BALD and ADV but worse than Margin and Entropy). On the contrary, KAL performs much better than KCENTER and KMEANS. In the *CUB200* task instead, 311 rules are employed in the KAL strategy considering bird species and their attributes (e.g., $\mathbf{White_Pelican} \Rightarrow \mathbf{Black_Eye} \vee \mathbf{Solid_Belly_Pattern} \vee \mathbf{Solid_Wing_Pattern}$), resulting in the top-two performances again, followed by Entropy. A separate comment is required instead for SupLoss, which provide low performances in the computer vision problems. Even in the optimal case where one is capable of perfectly predicting the supervision loss (as simulated here), selecting samples on which the network is wrong (i.e., with high supervision loss) might not be an optimal active strategy, since it might mostly select outliers. These results prove that exploiting symbolic knowledge is an efficient active learning strategy.

3.2 Ablation Studies

The amount of knowledge is directly proportional to the performance improvement To further show the importance of having a diverse and rich set of rules as introduced in Section 3.1, we performed here an ablation study. In particular, Table 2 reports the performance of the network when equipped with a KAL strategy considering only 0%, 25%, 50%, 75% or 100% of the available knowledge. The results show evidently that the amount of knowledge is directly proportional to the performance improvement, up to $+1.8\%$. In the 0% scenario, the only rule employed is the uncertainty-like rule, which was always retained. Notice how 50.13 is actually similar to the LeastConf result (50.20), suggesting that KAL without any further knowledge results in an uncertainty-based strategy. Furthermore, there might exist better uncertainty-like rules, which we plan to study in future work.

Table 2: Ablation study on the quantity of knowledge employed to support the KAL strategy on the *CUB200* dataset. The amount of knowledge is directly proportional to the increase of performance.

Dataset	KAL 0%	KAL 25%	KAL 50%	KAL 75%	KAL 100%
CUB200	50.13 ± 0.39	50.19 ± 0.42	50.22 ± 0.40	51.28 ± 0.29	51.98 ± 0.35

Selecting diverse constraint violations and uncertainty-like rule improve the performances Given a set of rules $\mathcal{K}$, the proposed method might in theory select p samples all violating the same rule $\phi_k(f(x))$. However, picking a batch of samples belonging to the same distribution might slow down training convergence. To avoid this issue, we select a maximum number r of samples violating a certain rule k, similarly to (Brinker, 2003) introducing diversity in margin-based approaches. Specifically, we group samples $x \in X_u$ according to the rule they violate the most, and we allow a maximum number of $p/2$ samples from each group (still following the ranking given by Eq. 1). In Appendix A.4, we report a table showing how requiring samples violating diverse constraints improves the overall quality of the KAL selection process. Also, we show the importance of adding the uncertainty-like rule $\bigwedge_i \mathbf{f_i} \oplus \neg \mathbf{f_i}$ introduced at the end of Section 2.3. Together, these two features allow improving the average performances of the network up to 2 %.

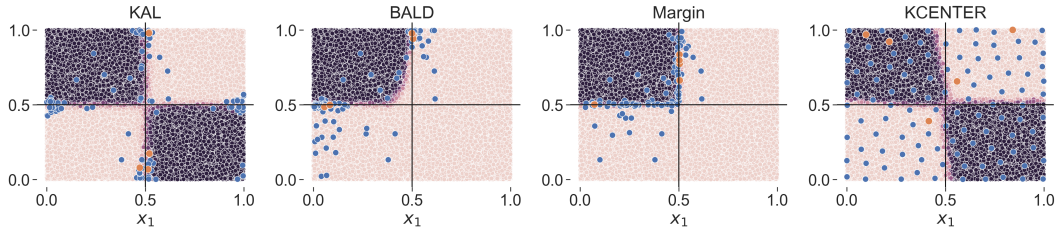

Figure 3: A comparison of the sample selection process on the *XOR-like* task at the 20th iteration (starting from the same points as in Figure 1). Notice how both uncertainty-based strategies (BALD and Margin) have not covered yet the novel data distribution (right-bottom quadrant).

3.3 KAL DISCOVERS NOVEL DATA DISTRIBUTIONS UNLIKE UNCERTAINTY STRATEGIES

For a better understanding of the results obtained, in Figure 3 we report the samples selected by some of the compared strategies after 20 iteration on the *XOR-like* task (around ~ 100 labelled data), starting from the same randomly selected samples of Figure 1. As already introduced in Section 2.2, the KAL strategy enables discovering data distribution lying far from the starting training data. Indeed, in the leftmost figure of Figure 3, we can appreciate how the network has fully learnt the data distribution in the right-bottom quadrant, even though it was not represented by the starting samples. On the contrary, when employing an uncertainty-based strategy (like Margin but also BALD, central figures), the consistent population of labelled data is mostly required along the decision boundary of the top-left distribution, with the bottom-right one that is yet to be discovered. The CoreSet representative strategy, instead, has covered the four quadrants. However, by only working on input features statistics and without notion on the predictions, this strategy does not choose points along the decision boundaries, preventing the network from reaching high accuracy performances. More figures showing the predictions at different iterations for all the strategies are reported in Appendix A.5.

3.4 KAL ENSURES DOMAIN EXPERTS THAT THEIR KNOWLEDGE IS ACQUIRED BY THE MODEL

It may be the case that domain experts are provided with a small corpus of rules which is crucial to be respected by the trained model, because, e.g., it has to be deployed in a sensitive context. By always selecting the data that violate this corpus of rules, KAL ensures them that the trained model respects the provided knowledge also at test time. To simulate this scenario, we computed the argument of Eq. 1 over a small part of the CUB knowledge $\mathcal{K}_{CUB-S}$ on the test data X_T for the f^b model trained with all the budget: $\varphi(\mathcal{K}_{CUB-S}, f^b, X_T) = \sum_{x \in X_T} \sum_{k \in \mathcal{K}_{CUB-S}} \varphi_k(f^b(x))$. In Table 3 we report the increased percentage of the violation by models trained following some of the compared methods w.r.t. the violation of a model trained following the KAL strategy and equipped with such corpora (KAL_{small}). The complete table together with more experimental detail is reported in appendix A.6. For the sake of completeness, this model reaches a lower test F1 AUBC (49.04). Nonetheless, it ensures domain experts that the provided knowledge is respected significantly more than using Random selection, or, worse, than standard active learning strategies.

Table 3: Violation of the $\mathcal{K}_{CUB-S}$ knowledge computed as the increased percentage over the violation of a model trained to respect this knowledge (KAL_{small}). The lower, the better. The proposed method ensures domain experts that their knowledge is respected by the model on test data.

KAL_{small}	Random	ADV_{BIM}	BALD	Entropy	KCENTER	LeastConf
+0.00%	+483.10%	+591.03%	+720.25	+861.74%	+530.13%	+1334.50%

3.5 KAL CAN BE EMPLOYED EVEN WHEN DOMAIN-KNOWLEDGE IS NOT AVAILABLE

It might be argued that the proposed strategy can be employed only when a domain knowledge is available. However, recent works in the eXplainable AI (XAI) field (Guidotti et al., 2018; Ribeiro et al., 2018; Ciravegna et al., 2020) have shown that we can extract the same knowledge from a trained model. In general, they achieve this by training a white-box model (e.g. a decision tree) to

Table 4: Accuracy of the KAL strategy coupled with a XAI method, extracting the knowledge from the same network. Notice how the AUCB reduction of performance is always smaller than 1 %.

	XOR	IRIS	Animals	CUB
KAL	97.73 ± 0.83	93.60 ± 3.87	56.15 ± 1.04	51.98 ± 0.35
KAL_{XAI}	97.11 ± 0.60	92.97 ± 4.48	55.71 ± 1.24	51.38 ± 0.29

globally explain the behaviour of a neural network. Here, we propose to employ these FOL-based explanations ($\mathcal{K}_{XAI}$) as the base knowledge of the proposed strategy when no other knowledge is available (KAL_{XAI}). More precisely, after each iteration, we employ a simple decision tree as proposed in Guidotti et al. (2018) to extract the knowledge. More details on how we trained the XAI method are reported in Appendix A.7. However, the knowledge may be partial, particularly during the first iterations, since it is extracted on the training distribution only. Therefore, we use Eq. 1 to select only 60% of the samples, with the remaining randomly selected. This allows to eventually recover the complete knowledge. In Table 4, we report the performance of the network when equipped with this strategy (KAL_{XAI}), together with the performance of the standard strategy. Notice how the reduction of performance is limited to less than 1%, confirming the validity of the proposed approach even in this scenario. The amount of randomly chosen samples has not been cross-validated, therefore we expect to get even higher results by fine-tuning this parameter.

3.6 KAL CAN BE EMPLOYED IN OBJECT RECOGNITION TASKS

To test the proposed method in an object recognition context, as a proof of concept, we experimented on the simple *DOGvsPERSON* dataset. On this task, we compute the AUBC of the mean Average Precision curves. Also in this case, the network increases more its performances when equipped with the KAL strategy (55.90 ± 0.39) with respect to standard random sampling (51.41 ± 1.25) but also compared to the SupLoss method (55.30 ± 0.54), proving the efficacy of the KAL strategy also in this context. A figure showing the three budget curves is reported in appendix A.8. Reported results are averaged over 3 initialization seeds. We only compared with Random selection and the simplified version of Yoo & Kweon (2019), since uncertainty-based strategies are not straightforward to apply in this context (Haussmann et al., 2020). Finally, we highlight again that the SupLoss performance reported is an upper bound of the performance of the method proposed in Yoo & Kweon (2019). Particularly in this context, we believe that the object recognition loss might not be easily learnt by an external model, reducing the performance of the SupLoss method.

3.7 KAL IS NOT COMPUTATIONALLY EXPENSIVE

When devising novel active learning techniques, of crucial importance is also the computational effort. Indeed, since re-training a deep neural network already requires a substantial amount of resources, the associated active strategy should be as light as possible. In Zhan et al. (2022), authors used as a term of comparison the average time needed to randomly sample a novel batch of data. In Table 5, we report the increased computational time in percentage. The KAL strategy (in its vanilla version) does not increase it significantly (with an average + 2-23 %). On the contrary, BALD (+ 44-191%) and, more importantly, clustering (+ 8-742%), CoreSet (2-5446%) and ADV-based (162-2451%) strategies demand considerable amounts of computational resources, strongly reducing the usability of the same methods. Standard uncertainty-based techniques like Margin, instead, are not computationally demanding, with only the Dropout versions having an increase of 2-80% (similarly to KAL_D). The complete table with all methods is reported in Appendix A.9.

4 RELATED WORK

Devising an active learning strategy is not an easy task. In the literature, two main approaches have been followed: uncertainty sampling which selects the data on which the model is the least confident; curriculum learning which instead focuses first on easy samples then extending the training set to incorporate more and more difficult ones while also targeting more diversity. Standard uncertain strategies consist in choosing samples associated to maximal prediction entropy (Houlsby

Table 5: Computational demand computed as the percentage increase over the time required for random sampling as defined in (Zhan et al., 2022). The lower, the better. Notice how the proposed method is less computationally expensive than many recent methods (ADV, BALD, KCENTER).

	XOR	IRIS	Animals	CUB
ADV_{BIM}	$+940.38\% \pm 278.11$	$+162.41\% \pm 31.15$	$+1251.36\% \pm 39.03$	$+1353.64\% \pm 33.53$
BALD	$+191.44\% \pm 15.14$	$+72.08\% \pm 47.00$	$+44.48\% \pm 3.34$	$+112.47\% \pm 13.28$
KCENTER	$+321.06\% \pm 48.54$	$+2.01\% \pm 1.92$	$+606.94\% \pm 56.57$	$+5446.31\% \pm 59.38$
KMEANS	$+8.30\% \pm 12.39$	$+9.39\% \pm 3.16$	$+173.20\% \pm 9.80$	$+742.47\% \pm 54.30$
Margin	$+0.66\% \pm 14.53$	$+1.85\% \pm 2.56$	$+1.26\% \pm 2.39$	$+0.65\% \pm 0.19$
Margin_D	$+84.00\% \pm 13.94$	$+1.69\% \pm 3.90$	$+3.95\% \pm 3.93$	$+38.88\% \pm 2.49$
KAL	$+2.52\% \pm 13.69$	$+0.32\% \pm 2.56$	$+6.43\% \pm 2.02$	$+23.42\% \pm 0.21$
KAL_D	$+88.28\% \pm 20.79$	$+5.41\% \pm 6.23$	$+8.18\% \pm 1.29$	$+62.98\% \pm 2.37$

et al., 2011) or at minimum distance from the hyperplane in SVM (Schohn & Cohn, 2000) or with the highest variation ratio in Query-by-committee with ensemble methods (Ducoffe & Precioso, 2017; Beluch et al., 2018). Establishing prediction uncertainty is more difficult with DL models. Indeed, they generally tend to be over-confident, particularly when employing softmax activation functions (Thulasidasan et al., 2019). Furthermore, there is no easy access to the distance to the decision boundary as for SVM, so it needs to be computed. This problem has been tackled by devising different uncertain strategies, such as employing Bayesian Neural Network with Monte Carlo Dropout (Gal et al., 2017), by calculating the minimum distance required to create an adversarial example (Ducoffe & Precioso, 2018), or even predicting the loss associated to unlabelled sample (Yoo & Kweon, 2019). As pointed out by (Pop & Fulop, 2018), however, uncertain strategy may choose the same categories many times and create unbalanced datasets. To solve this, uncertain sample selection can be coupled with diversity sampling strategies. Diversity is generally obtained by preferring batches of data maximizing the mutual information between model parameters and predictions (Kirsch et al., 2019), or core-set points (Sener & Savarese, 2018), or, also, samples nearest to k-means cluster centroids (Zhdanov, 2019).

It has been pondered that human cognition mainly consists in two different tasks: perceiving the world and reasoning over it (Solso et al., 2005). While in humans they take place at the same times, in artificial intelligence these two tasks are separately conducted by machine learning and logic programming. It has been argued that joining these tasks (to create a so-called hybrid model) may overcome some of the most important limits of deep learning, among which the *"data hungry"* issue (Marcus, 2018). In the literature, there exists a variety of proposals aiming at this objective, ranging from Statistical Relational Learning (SRL) (Koller et al., 2007) and Probabilistic Logic Programming (De Raedt & Kimmig, 2015) which focuses on integrating learning with logic reasoning, to enhanced networks focusing on relations or with external memories (Santoro et al., 2017; Graves et al., 2016). The learning from constraints framework (Gnecco et al., 2015; Diligenti et al., 2017), instead, computes and enforce the satisfaction of a given domain knowledge within DL models. In this paper, we have shown that it can be naturally leveraged to devise active learning strategies, providing the first example of integration of symbolic knowledge in this context.

5 Conclusions

In this paper, we proposed an active learning strategy driven by knowledge consistency principles. It inspects model predictions on unseen data to detect those violating the logic constraints derived from a domain knowledge. The performance of a model equipped with such a strategy outperforms standard uncertainty-based approaches, without being computationally demanding. Furthermore, we think that KAL could induce more trust in DL with respect to standard active learning techniques by enabling non-expert users to train models leveraging their domain knowledge and ensuring them that it will be respected. One may argue that the main limitation of the proposed approach is in computer vision contexts, if no attributes about the main classes are known. A possible solution could be to automatically extract such concepts from the latent space of the network, as recently proposed in several works (Ghorbani et al., 2019; Chen et al., 2020).

REFERENCES

Pietro Barbiero, Gabriele Ciravegna, Francesco Giannini, Pietro Lió, Marco Gori, and Stefano Melacci. Entropy-based logic explanations of neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pp. 6046–6054, 2022.

William H Beluch, Tim Genewein, Andreas Nürnberger, and Jan M Köhler. The power of ensembles for active learning in image classification. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 9368–9377, 2018.

Christopher M Bishop. *Pattern Recognition and Machine Learning*, volume 4. Springer, 2006.

K. Brinker. Incorporating diversity in active learning with support vector machines. In *ICML*, pp. 59–66, 2003.

Zhi Chen, Yijie Bei, and Cynthia Rudin. Concept whitening for interpretable image recognition. *Nature Machine Intelligence*, 2(12):772–782, dec 2020. doi: 10.1038/s42256-020-00265-z. URL `https://doi.org/10.1038%2Fs42256-020-00265-z`.

Gabriele Ciravegna, Francesco Giannini, Marco Gori, Marco Maggini, and Stefano Melacci. Human-driven fol explanations of deep learning. In *IJCAI*, pp. 2234–2240, 2020.

Arun Das and Paul Rad. Opportunities and challenges in explainable artificial intelligence (xai): A survey. *arXiv preprint arXiv:2006.11371*, 2020.

L. De Raedt and A. Kimmig. Probabilistic (logic) programming concepts. *Machine Learning*, 100 (1):5–47, 2015.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

M. Diligenti, M. Gori, and C. Sacca. Semantic-based regularization for learning and inference. *Artificial Intelligence*, 244:143–165, 2017.

M. Ducoffe and F. Precioso. Active learning strategy for cnn combining batchwise dropout and query-by-committee. In *ESANN*, 2017.

M. Ducoffe and F. Precioso. Adversarial active learning for deep networks: a margin based approach. *arXiv:1802.09841*, 2018.

Yarin Gal, Riashat Islam, and Zoubin Ghahramani. Deep bayesian active learning with image data. In *International Conference on Machine Learning*, pp. 1183–1192. PMLR, 2017.

Amirata Ghorbani, James Wexler, James Y Zou, and Been Kim. Towards automatic concept-based explanations. *Advances in Neural Information Processing Systems*, 32, 2019.

Leilani H Gilpin, David Bau, Ben Z Yuan, Ayesha Bajwa, Michael Specter, and Lalana Kagal. Explaining explanations: An overview of interpretability of machine learning. In *2018 IEEE 5th International Conference on data science and advanced analytics (DSAA)*, pp. 80–89. IEEE, 2018.

G. Gnecco, M. Gori, S. Melacci, and M. Sanguineti. Foundations of support constraint machines. *Neural computation*, 27(2):388–480, 2015.

Alex Graves, Greg Wayne, Malcolm Reynolds, Tim Harley, Ivo Danihelka, Agnieszka Grabska-Barwinska, Sergio Gomez Colmenarejo, Edward Grefenstette, Tiago Ramalho, John P. Agapiou, Adrià Puigdomènech Badia, Karl Moritz Hermann, Yori Zwols, Georg Ostrovski, Adam Cain, Helen King, Christopher Summerfield, Phil Blunsom, Koray Kavukcuoglu, and Demis Hassabis. Hybrid computing using a neural network with dynamic external memory. *Nature*, 538:471–476, 2016.

Riccardo Guidotti, Anna Monreale, Salvatore Ruggieri, Dino Pedreschi, Franco Turini, and Fosca Giannotti. Local rule-based explanations of black box decision systems. *arXiv preprint arXiv:1805.10820*, 2018.

E. Haussmann, M. Fenzi, K. Chitta, J. Ivanecky, H. Xu, D. Roy, A. Mittel, N. Koumchatzky, C. Farabet, and J.M. Alvarez. Scalable active learning for object detection. In *IEEE IV Symposium*, pp. 1430–1435. IEEE, 2020.

N. Houlsby, F. Huszár, Z. Ghahramani, and M. Lengyel. Bayesian active learning for classification and preference learning, 2011.

Kuan-Hao Huang. Deepal: Deep active learning in python. *arXiv preprint arXiv:2111.15258*, 2021.

A. Kirsch, J. Van Amersfoort, and Y. Gal. Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning. *NeurIPS*, 32:7026–7037, 2019.

E.P. Klement, R. Mesiar, and E. Pap. *Triangular norms*, volume 8. Springer Science & Bus. Media, 2013.

P. W. Koh, T. Nguyen, Y. S. Tang, S. Mussmann, E. Pierson, B. Kim, and P. Liang. Concept bottleneck models. In *ICML*, pp. 5338–5348, 2020.

D. Koller, N. Friedman, S. Džeroski, C. Sutton, A. McCallum, A. Pfeffer, P. Abbeel, M.-F. Wong, C. Meek, J. Neville, et al. *Introduction to statistical relational learning*. MIT press, 2007.

Y. LeCun, Y. Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

I. Loshchilov and F. Hutter. Decoupled weight decay regularization. *arXiv:1711.05101*, 2017.

G. Marcus. Deep learning: A critical appraisal. *arXiv:1801.00631*, 2018.

G. Marra, F. Giannini, M. Diligenti, and M. Gori. Lyrics: A general interface layer to integrate logic inference and deep learning. In *ECML/PKDD*, 2019.

A. K. McCallumzy and K. Nigamy. Employing em and pool-based active learning for text classification. In *ICML*, pp. 359–367. Citeseer, 1998.

S. Melacci, G. Ciravegna, A. Sotgiu, A. Demontis, B. Biggio, M. Gori, and F. Roli. Domain knowledge alleviates adversarial attacks in multi-label classifiers. *IEEE PAMI*, pp. 1–1, 2021. doi: 10.1109/TPAMI.2021.3137564.

Yuval Netzer, Tao Wang, Adam Coates, Alessandro Bissacco, Bo Wu, and Andrew Y. Ng. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning 2011*, 2011. URL `http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf`.

R. Pop and P. Fulop. Deep ensemble bayesian active learning: Addressing the mode collapse issue in monte carlo dropout via ensembles. *arXiv:1811.03897*, 2018.

J. Redmon and A. Farhadi. Yolo v3: An incremental improvement. *CoRR*, abs/1804.02767, 2018. URL `http://arxiv.org/abs/1804.02767`.

Pengzhen Ren, Yun Xiao, Xiaojun Chang, Po-Yao Huang, Zhihui Li, Brij B Gupta, Xiaojiang Chen, and Xin Wang. A survey of deep active learning. *ACM Computing Surveys (CSUR)*, 54(9):1–40, 2021.

Marco Tulio Ribeiro, Sameer Singh, and Carlos Guestrin. Anchors: High-precision model-agnostic explanations. In *Proceedings of the AAAI conference on artificial intelligence*, volume 32, 2018.

A. Santoro, D. Raposo, D. G. Barrett, M. Malinowski, R. Pascanu, P. Battaglia, and T. Lillicrap. A simple neural network module for relational reasoning. *NeurIPS*, 30, 2017.

G. Schohn and D. Cohn. Less is more: Active learning with support vector machines. *ICML*, 10 2000.

O. Sener and S. Savarese. Active learning for convolutional neural networks: A core-set approach. In *ICLR*, 2018.

Ozan Sener and Silvio Savarese. Active learning for convolutional neural networks: A core-set approach. *arXiv preprint arXiv:1708.00489*, 2017.

L. Serafini and A. d'Avila Garcez. Logic tensor networks: Deep learning and logical reasoning from data and knowledge. *arXiv:1606.04422*, 2016.

Burr Settles. Active learning literature survey. 2009.

R.L. Solso, M.K. MacLin, and O.H. MacLin. *Cognitive psychology*. Pearson Education New Zealand, 2005.

S. Thulasidasan, G. Chennupati, J.A. Bilmes, T. Bhattacharya, and S. Michalak. On mixup training: Improved calibration and predictive uncertainty for deep neural networks. *NeurIPS*, 32, 2019.

C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The Caltech-UCSD Birds-200-2011 Dataset. Technical Report CNS-TR-2011-001, CalTech, 2011.

Dan Wang and Yi Shang. A new active labeling method for deep learning. In *2014 International joint conference on neural networks (IJCNN)*, pp. 112–119. IEEE, 2014.

P.H. Winston and B.K. Horn. Lisp. 1986.

D. Yoo and I.S. Kweon. Learning loss for active learning. In *IEEE CVPR*, pp. 93–102, 2019.

F. Yu, A. Seff, Y. Zhang, S. Song, T. Funkhouser, and J. Xiao. Lsun: Construction of a large-scale image dataset using deep learning with humans in the loop. *arXiv:1506.03365*, 2015.

Xueying Zhan, Huan Liu, Qing Li, and Antoni B. Chan. A comparative survey: Benchmarking for pool-based active learning. In Zhi-Hua Zhou (ed.), *Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21*, pp. 4679–4686. International Joint Conferences on Artificial Intelligence Organization, 8 2021. doi: 10.24963/ijcai.2021/634. URL https://doi.org/10.24963/ijcai.2021/634. Survey Track.

Xueying Zhan, Qingzhong Wang, Kuan-hao Huang, Haoyi Xiong, Dejing Dou, and Antoni B. Chan. A comparative survey of deep active learning, 2022. URL https://arxiv.org/abs/2203.13450.

F. Zhdanov. Diverse mini-batch active learning. *arXiv:1901.05954*, 2019.

X. Zhu and A.B. Goldberg. Introduction to semi-supervised learning. *Synthesis lectures on artificial intelligence and machine learning*, 3(1):1–130, 2009.

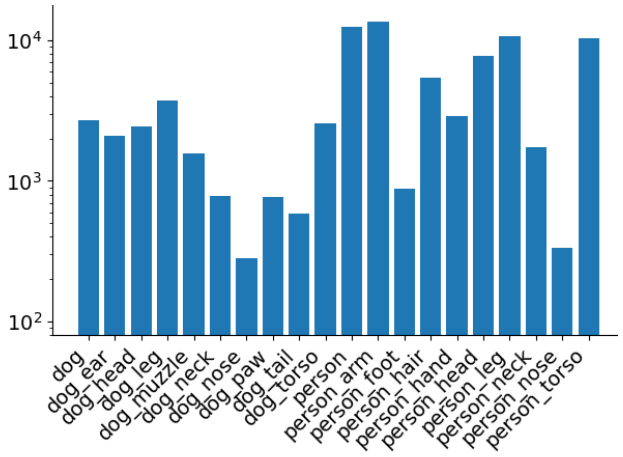

Figure 4: Class distribution in the *DOGvsPERSON* dataset.

A EXPERIMENTAL DETAILS AND FURTHER RESULTS

A.1 THE *DOGvsPERSON* DATASET

The *DOGvsPERSON* dataset is a publicly available dataset that we have created for showing the potentiality of the proposed method on a simple object-recognition problem, where well-defined relations are present among the classes, and we can employ common-knowledge rule to easily relate them. It is extracted from the *PASCAL-Part* dataset by considering only the Dog and Person main classes and their corresponding parts. In the original *PASCAL-Part* dataset, labels are given in the form of segmentation masks. We extracted a bounding box from each mask by considering the leftmost and highest pixel as the first coordinate and the rightmost and lowest pixel as the second one. Very specific parts are merged into a single class, following the approach of Serafini & d'Avila Garcez (2016) (e.g., **Left_Lower_Arm**, **Left_Upper_Arm**, **Right_Lower_Arm**, **Right_Upper_Arm** becomes **Arm**). Differently from the standard *PASCAL-Part* dataset, however, the parts in common to different objects are considered as different classes (**head** becomes **Dog_Head**, **Person_Head**). Furthermore, we only consider masks having areas $\geq 1\%$ of the whole image areas as valid label. At last, only classes appearing at least 100 times are retained. This lead to a total of 20 classes with 2 main classes (**Dog** and **Person**) and 18 parts (**Dog_Ear**, **Dog_Head**, **Dog_Leg**, **Dog_Muzzle**, **Dog_Neck**, **Dog_Nose**, **Dog_Paw**, **Dog_Tail**, **Dog_Torso**, **Person_Arm**, **Person_Foot**, **Person_Hair**, **Person_Hand**, **Person_Head**, **Person_Leg**, **Person_Neck**, **Person_Nose**, **Person_Torso**) displayed in a total of 4304 samples. Final classes are distributed in the samples as shown in Figure 4.

A.2 NETWORK ARCHITECTURES, HYPERPARAMETERS, AND DOMAIN KNOWLEDGES

XOR-like The problem of inferring the *XOR-like* operation has been already introduced in Section 2.2: it is an artificial dataset consisting of 100000 samples $x \in X \subset R^2$, mapped to the corresponding label $y \in Y \subset [0, 1]$ as in Eq. 2. A Multi-Layer Perceptron (MLP) $f\colon X \to Y$ is used to solve the task[2]. It is equipped with a single hidden layer of 100 neurons and Rectified Linear Unit (ReLU) activation, and a single output neuron with sigmoid activation. It has been trained with an AdamW optimizer Loshchilov & Hutter (2017) for 250 epochs at each iteration, with a learning rate $\eta = 10^{-3}$. Standard cross-entropy loss has been used to enforce f to learn the available supervisions. By starting from $n = 10$ samples, we added $p = 5$ labelled samples at each iteration for a total of $q = 78$ iterations, resulting in a total final budget of $b = 400$ labelled samples. As anticipated, in the *XOR* problem, the rule employed for the KAL strategy is $\forall x\ \mathbf{x_1} \oplus \mathbf{x_2} \Leftrightarrow \mathbf{f}$, as also reported in Table 6.

[2]For all strategies requiring a multi-class output, we considered the network output as $\hat{f} = [f, 1 - f]$.

Table 6: Domain knowledge on the *XOR-like* dataset.

$\forall x \quad \mathbf{x_1} \oplus \mathbf{x_2} \Leftrightarrow \mathbf{f}$

For more real-life style problems, we have considered four more datasets where the domain-knowledge is partial or only related to the class functions.

IRIS The *IRIS*[3] dataset is the standard iris-species classification problem. More precisely, the task consists in classifying $c = 3$ Iris species (Iris Setosa, Iris Versicolour, Iris Virginica) starting from $d = 4$ features (sepal length, sepal width, petal length, petal width). To solve the learning problem, an MLP $f\colon X^d \rightarrow Y^c$ is employed, with one hidden layer composed of 100 neurons equipped with ReLu activation functions. It has been trained again with AdamW optimizer for 200 epochs at each iteration and learning rate $\eta = 3 * 10^{-3}$. A cross-entropy loss is employed to enforce the supervisions. By starting from $n = 5$ points and by adding $p = 5$ labelled samples at each iteration for $q = 14$ iterations, leading to a total budget of $b = 75$ labelled samples. The knowledge employed in this case consists of 3 very simple rules (one per class) based on the two predicates $\mathbf{Long_Petal}$ and $\mathbf{Wide_Petal}$ (built on the 3rd and 4th features of the dataset, respectively, as explained in Section 2.2). In addition, in this case, a mutually exclusive rule on the classes is also considered $\forall x, \mathbf{Setosa} \oplus \mathbf{Versicolour} \oplus \mathbf{Virginica}$, as reported in Table 7.

Table 7: Domain knowledge on the *IRIS* dataset.

$\forall x$	$\neg$ Long_Petal $\Leftrightarrow$ Setosa
$\forall x$	Long_Petal $\wedge \neg$ Wide_Petal $\Leftrightarrow$ Versicolour
$\forall x$	Long_Petal $\wedge$ Wide_Petal $\Leftrightarrow$ Virginica
$\forall x$	Setosa $\oplus$ Versicolour $\oplus$ Virginica

ANIMALS The *Animals*' dataset is a collection of 8287 images of animals, taken from the ImageNet database[4]. The task consists in the classification of 7 main classes ($\mathbf{Albatross}, \mathbf{Giraffe}, \mathbf{Cheetah}, \mathbf{Ostrich}, \mathbf{Penguin}, \mathbf{Tiger}, \mathbf{Zebra}$) and 26 animal attributes (e.g., $\mathbf{Mammal}, \mathbf{Fly}$ or $\mathbf{Lay_Eggs}$), for a total of $c = 33$ classes. In this case, a Resnet50 CNN has been employed to solve the task $f\colon X^d \rightarrow Y^c$. Going into more details, a transfer learning strategy has been employed: the network f has been pretrained on the ImageNet dataset (Deng et al., 2009), and two fully connected layers (the first one equipped with 100 neurons) have been trained (from scratch) on the *ANIMALS* dataset. Again, an AdamW optimizer is considered with a learning rate $\eta = 10^{-3}$ employed for 250 epochs of training at each iteration, with binary cross-entropy loss since we deal with a multi-label problem. We started with $n = 100$ labelled samples, and we added $p = 50$ samples each time for $q = 48$ iterations, for a final budget of $b = 2500$ labelled samples. In the case of *Animals*, the employed knowledge is a simple collection of 16 FOL formulas, defined by Winston & Horn (1986) as a benchmark. They involve relationships between animals and their attributes, such as $\forall x \mathbf{Fly} \wedge \mathbf{Lay_Eggs} \Rightarrow \mathbf{Bird}$. To this collection of rules, we have also added a mutual exclusive disjunction among the animal classes (only one animal is present in each image) and a standard disjunction over the animal attributes (each animal may be associated to many attributes). The complete list of rules employed is reported in Table 8.

CUB200 The Caltech-UCSD Birds-200-2011[5] dataset Wah et al. (2011) is a collection of 11,788 images of birds. The task consists in the classification of 200 birds species (e.g., $\mathbf{Black_foooted_Albatross}$) and birds attributes (e.g., $\mathbf{White_Throat}, \mathbf{Medium_Size}$). Attribute annotation, however, is quite noisy. For this reason, attributes are denoised by considering

[3] *Iris*: `https://archive.ics.uci.edu/ml/datasets/iris`

[4] *Animals (Imagenet)*: `http://www.image-net.org/`, released with BSD 3-Clause "New" or "Revised" License

[5] *CUB200*:`http://www.vision.caltech.edu/visipedia/CUB-200-2011` released with MIT License.

Table 8: Domain knowledge on the *Animals* dataset.

$\forall x$	Hair $\vee$ Mammal
$\forall x$	Milk $\Rightarrow$ Mammal
$\forall x$	Feather $\Rightarrow$ Bird
$\forall x$	Fly $\wedge$ LayEggs $\Rightarrow$ Bird
$\forall x$	Mammal $\wedge$ Meat $\Rightarrow$ Carnivore
$\forall x$	Mamal $\wedge$ PointedTeeth $\wedge$ Claws $\wedge$ ForwardEyes $\Rightarrow$ Carnivore
$\forall x$	Mammal $\wedge$ Hoofs $\Rightarrow$ Ungulate
$\forall x$	Mammal $\wedge$ Cud $\Rightarrow$ Ungulate
$\forall x$	Mammal $\wedge$ Cud $\Rightarrow$ Eventoed
$\forall x$	Carnivore $\wedge$ Tawny $\wedge$ DarkSpots $\Rightarrow$ Cheetah
$\forall x$	Carnivore $\wedge$ Tawny $\wedge$ BlackStripes $\Rightarrow$ Tiger
$\forall x$	Ungulate $\wedge$ LongLegs $\wedge$ LongNeck $\wedge$ Tawny $\wedge$ DarkSpots $\Rightarrow$ Giraffe
$\forall x$	Blackstripes $\wedge$ Ungulate $\wedge$ White $\Rightarrow$ Zebra
$\forall x$	Bird $\wedge$ $\neg$Fly $\wedge$ LongLegs $\wedge$ LongNeck $\wedge$ Black $\Rightarrow$ Ostrich
$\forall x$	Bird $\wedge$ $\neg$Fly $\wedge$ Swim $\wedge$ BlackWhite $\Rightarrow$ Penguin
$\forall x$	Bird $\wedge$ GoodFlier $\Rightarrow$ Albatross
$\forall x$	Albatross $\oplus$ Giraffe $\oplus$ Cheetah $\oplus$ Ostrich $\oplus$ Penguin $\oplus$ Tiger $\oplus$ Zebra)
$\forall x$	Mammal $\vee$ Hair $\vee$ Milk $\vee$ Feathers $\vee$ Bird $\vee$ Fly $\vee$ Meat $\vee$ Carnivore $\vee$ PointedTeeth $\vee$ Claws $\vee$ ForwardEyes $\vee$ Hoofs $\vee$ Ungulate $\vee$ Cud $\vee$ Eventoed $\vee$ Tawny $\vee$ BlackStripes $\vee$ LongLegs $\vee$ LongNeck $\vee$ DarkSpots $\vee$ White $\vee$ Black $\vee$ Swim $\vee$ BlackWhite $\vee$ GoodFlier

class-level annotations similarly to Koh et al. (2020). A certain attribute is set as present only if it is also present in at least 50 images of the same class. Furthermore, we only considered attributes present in at least 10 classes after this refinement. In the end, 108 attributes have been retained, for a total of $c = 308$ classes. Images have been resized to a dimension $d = 256 \times 256$ pixels. The same network as in the *ANIMALS* case has been employed to solve the learning problem, with two fully connected layers trained from scratch (the first one equipped with 620 neurons – twice the dimension of the following layer). Again, an AdamW optimizer is considered with a learning rate $\eta = 10^{-3}$ for 100 epochs of training. Owing to the increased difficulty of the problem, we started with $n = 2000$ labelled samples, and we added $p = 200$ samples for $q = 25$ iterations, for a final budget of $b = 7000$ labelled samples. The knowledge employed in this case consider the relation between the classes and their attributes, with logic implications both from the class to the attributes (e.g., $\mathbf{White_Pelican} \Rightarrow \mathbf{Black_Eye} \vee \mathbf{Solid_Belly_Pattern} \vee \mathbf{Solid_Wing_Pattern}$), and the vice-versa (e.g., $\mathbf{Striped_Breast_Pattern} \Rightarrow \mathbf{Parakeet_Auklet} \vee \mathbf{Black_throated_Sparrow} \vee \ldots$) Furthermore, a disjunction on the main classes[6] ($\mathbf{Black_footed_Albatross} \vee \mathbf{Laysan_Albatross} \vee \mathbf{Sooty_Albatross} \vee \ldots$) and one on the attributes are considered ($\mathbf{Dagger_Bill} \vee \mathbf{Hooked_Bill} \vee \mathbf{All_Purpose_Bill} \vee \mathbf{Cone_Bill} \vee \ldots$). A few examples of the rules employed are reported in Table 9.

DOGvsPERSON This dataset has already been introduced in Appendix A.1. Since we filtered out very small object masks in this dataset, we have been able to employ a YOLOv3 model Redmon & Farhadi (2018) to solve the object-recognition problem. The model has been trained for 100 epochs at each iteration with an AdamW optimizer, with a learning rate $\eta = 3 * 10^{-4}$ decreasing by 1/3 every 33 epochs. For both training and evaluation, the Input Over Union (IOU) threshold has been set to 0.5, the confidence threshold to 0.01 and the Non-Maximum Suppression (NMS) threshold to 0.5. We started training with $n = 1000$ labelled examples and by adding $p = 500$ samples for $q = 4$ iterations for a final budget of $b = 2000$ labelled examples. In Section 3.6, we reported the AUBC of the mean Average Precision (mAP) of the model averaged 10 times with Intersection over Union (IoU) ranging from 0.5 to 0.95. On *DOGvsPERSON* we considered a set of rules listing the parts belonging to the dog or the person, (e.g., $\mathbf{Person} \Rightarrow \mathbf{Person_Arm} \vee \mathbf{Person_Foot} \vee \mathbf{Person_Hair} \vee \mathbf{Person_Hand} \vee \ldots$), the opposite rules implying the presence of the main object given the part (e.g., $\mathbf{Person_Foot} \Rightarrow \mathbf{Person}$). Also, we considered a disjunction of all the main classes and a disjunction of all the object-parts, for a total of 22 rules employed, as reported in Table 10.

[6]Due to the dimensionality of the dataset, the mutual exclusion of the main classes was computationally too expensive to compute in this case.

Table 9: Domain knowledge on the *CUB200* dataset. Only a few rules for each type have been reported for the sake of clarity. Also, due to the length of the rules, rules with more than two terms implied have been truncated.

$\forall x$	Black_footed_Albatross $\Rightarrow$ has_bill_shape_all-purpose $\wedge$ has_underparts_color_yellow $\wedge \ldots$
$\forall x$	Laysan_Albatross $\Rightarrow$ has_bill_shape_hooked_seabird $\wedge$ has_breast_pattern_solid $\wedge \ldots$
$\forall x$	Sooty_Albatross $\Rightarrow$ has_bill_shape_hooked_seabird $\wedge$ has_wing_color_black $\wedge \ldots$
$\forall x$	Groove_billed_Ani $\Rightarrow$ has_bill_shape_hooked_seabird $\wedge$ has_breast_pattern_solid $\wedge \ldots$
$\forall x$	Crested_Auklet $\Rightarrow$ has_wing_color_black $\wedge$ has_upperparts_color_black $\wedge \ldots$
	$\ldots$ $\ldots$
$\forall x$	has_bill_shape_dagger $\Rightarrow$ Green_Kingfisher $\wedge$ Pied_Kingfisher $\wedge \ldots$
$\forall x$	has_bill_shape_hooked_seabird $\Rightarrow$ Laysan_Albatross $\wedge$ Sooty_Albatross $\wedge \ldots$
$\forall x$	has_bill_shape_all-purpose $\Rightarrow$ Black_footed_Albatross $\wedge$ Red_winged_Blackbird $\wedge \ldots$
$\forall x$	has_bill_shape_cone $\Rightarrow$ Parakeet_Auklet $\wedge$ Indigo_Bunting $\wedge \ldots$
$\forall x$	has_wing_color_brown $\Rightarrow$ Brandt_Cormorant $\wedge$ American_Crow $\wedge \ldots$
	$\ldots$ $\ldots$
$\forall x$	Black_footed_Albatross $\vee$ Laysan_Albatross $\vee$ Sooty_Albatross $\vee \ldots$
$\forall x$	has_bill_shape_dagger $\vee$ has_bill_shape_hooked_seabird $\vee$ has_bill_shape_all-purpose $\vee \ldots$

Table 10: Domain knowledge on the *DOGvsPERSON* dataset.

$\forall x$	Dog_ear $\Rightarrow$ Dog
$\forall x$	Dog_head $\Rightarrow$ Dog
$\forall x$	Dog_leg $\Rightarrow$ Dog
$\forall x$	Dog_muzzle $\Rightarrow$ Dog
$\forall x$	Dog_neck $\Rightarrow$ Dog
$\forall x$	Dog_nose $\Rightarrow$ Dog
$\forall x$	Dog_paw $\Rightarrow$ Dog
$\forall x$	Dog_tail $\Rightarrow$ Dog
$\forall x$	Dog_torso $\Rightarrow$ Dog
$\forall x$	Person_arm $\Rightarrow$ Person
$\forall x$	Person_foot $\Rightarrow$ Person
$\forall x$	Person_hair $\Rightarrow$ Person
$\forall x$	Person_hand $\Rightarrow$ Person
$\forall x$	Person_head $\Rightarrow$ Person
$\forall x$	Person_leg $\Rightarrow$ Person
$\forall x$	Person_neck $\Rightarrow$ Person
$\forall x$	Person_nose $\Rightarrow$ Person
$\forall x$	Person_torso $\Rightarrow$ Person
$\forall x$	Dog $\Rightarrow$ Dog_ear $\vee$ Dog_head $\vee$ Dog_leg $\vee$ Dog_muzzle $\vee$ Dog_neck $\vee$ Dog_nose $\vee$ Dog_paw $\vee$ Dog_tail $\vee$ Dog_torso
$\forall x$	Person $\Rightarrow$ Person_arm $\vee$ Person_foot $\vee$ Person_hair $\vee$ Person_hand $\vee$ Person_head $\vee$ Person_leg $\vee$ Person_neck $\vee$ Person_nose $\vee$ Person_torso
$\forall x$	Dog $\vee$ Person
$\forall x$	Dog_ear $\vee$ Dog_head $\vee$ Dog_leg $\vee$ Dog_muzzle $\vee$ Dog_neck $\vee$ Dog_nose $\vee$ Dog_paw $\vee$ Dog_tail $\vee$ Dog_torso $\vee$ Person_arm $\vee$ Person_foot $\vee$ Person_hair $\vee$ Person_hand $\vee$ Person_head $\vee$ Person_leg $\vee$ Person_neck $\vee$ Person_nose $\vee$ Person_torso

In all experiments, we employed weight decay and low learning rate to avoid overfitting, rather than employing an early stopping strategy on a separate validation set. Indeed, we argue that it is not really realistic to rely on sufficiently large validation sets in an active learning scenario, where the amount of labels is scarce, and we try to minimize it as much as possible. Also, rather than retraining the network from scratch at each iteration, we choose to keep training it to save computational time.

Table 11: Resume of the active losses $\mathcal{L}_a(f, x)$ maximized by each of the compared methods. Variants of the same methods (e.g., employing Monte Carlo Dropouts) have not been reported for the sake of brevity. We indicate with H the Entropy, with σ the softmax activation, with $p_f(y_i, x)$ the probability of associated to the class i, with M the set of main classes, with ϵ the adversarial perturbation, $g(\cdot)$ the model used to learn the model loss, with Z_{CENTER} the CoreSet points and with Z_{MEANS} the centroids of K-Means.

Entropy	$\mathcal{L}_a = H[f(x)\|x] = -\left(\sigma(f_M(x)) \cdot \log(\sigma(f_M(x))\right)$
Margin	$\mathcal{L}_a = -\left(p_f(\hat{y}_1\|x) - p_f(\hat{y}_2\|x)\right), \quad \hat{y}_1, \hat{y}_2 = \arg\max^2_{i \in M} p_f(y_i, x)$
LeastConf	$\mathcal{L}_a = 1 - p_f(\hat{y}\|x), \quad \hat{y} = \arg\max_{i \in M} p_f(y_i, x)$
BALD	$\mathcal{L}_a = H[f(x)\|x] - \mathbb{E}[H[f(x)\|x, \omega]$
ADV	$\mathcal{L}_a = \|\|\epsilon\|\|, \quad \epsilon\colon f_M(x + \epsilon) \neq f_M(x)$
SupLoss	$\mathcal{L}_a = \mathcal{L}(f(x), y) \sim g(f(x))$
KCENTER	$\mathcal{L}_a = \|\|x - z_k\|\|, \quad z_k = \arg\min_{z \in Z_{CENTER}} \|\|x - z\|\|$
KMEANS	$\mathcal{L}_a = -\|\|x - z_k\|\|, \quad z_k = \arg\min_{z \in Z_{MEANS}} \|\|x - z\|\|$
KAL	$\mathcal{L}_a = \sum_{k \in \mathcal{K}} \varphi_k(f(x))$

A.3 Compared method details

In the experiments, several methods have been evaluated, comparing the performances both in terms of F1 score improvement and in terms of selection time. Compared techniques mostly follow two different active learning philosophies: uncertainty sample selection, which aim at estimating prediction uncertainty in deep neural networks; and diversity selection, which aims at maximally covering the input data distribution. Several standard strategies like **Entropy** (Settles, 2009), **Margin** (Netzer et al., 2011) and **LeastConf** (Wang & Shang, 2014) belong to the first group, as well as some more recent methods like **BALD** (Gal et al., 2017), **ADV**$_{DEEPFOOL}$ (Ducoffe & Precioso, 2018), **ADV**$_{BIM}$ (Zhan et al., 2022) and **SupLoss** (Yoo & Kweon, 2019). On the contrary, **KMeans** (Zhdanov, 2019) and KCenter (Sener & Savarese, 2017), aim at reaching the highest diversity among selected samples.

In Table 11, we reported the loss $\mathcal{L}_a(f, x)$ used by each method to select active samples as following:

$$x^\star = \arg\max_{x \in X_U} \mathcal{L}_a(f, x) \tag{4}$$

As the name implies, the **Entropy** method aims at measuring the uncertainty of a prediction by means of its entropy. With σ we indicate the softmax activation, that we applied on top[7] of f_M to obtain a probability distribution – i.e., $\sum_{i \in M} \sigma(f_i, x) = 1.$ – as required to compute the Entropy. To adapt this method to the multi-label context, we restricted the computation to the M main classes (mutually exclusive) of f. Similarly, in **Margin** and **LeastConf** we compute again the softmax over the main classes f_M to obtain for each main class i the probability $p_f(y_i, x)$ of being active. In the first case, the margin between the two classes associated to the highest probability is used to estimate the uncertainty of the prediction. In the second case, the inverse of the probability associated to the most probable class is used instead. **BALD** employs as acquisition function the mutual information between the model predictions and the model parameters. The first term of the loss is again the Entropy of the model predictions; the second term of the equation, instead, represents the expected value of the entropy over the posteriori of the model parameters. Basically, the overall value is high when the predictions differ a lot, but the confidence of each prediction is rather high. Both **ADV**$_{DEEPFOOL}$ and **ADV**$_{BIM}$ use as a metric of uncertainty the norm of ϵ, the minimum input alteration allowing to change the prediction of the network. The difference in the two methods relies on the way ϵ is computed (i.e., on the adversarial attack employed). The stopping criterion to find ϵ in both methods consists in finding a perturbation that induce the network to predict a different class. To adapt these methods to the multi-label context, therefore, we restricted again the classes to the main ones f_M. The **SupLoss** strategy, instead, aims at approximating the supervision loss by means of a model $g(f(x))$. It receives in input the output of the f network (as well as the activation of the last hidden layers) and is trained to mimic the actual loss $\mathcal{L}(f(x), y) \sim g(f(x))$ on

[7]To avoid numerical issues, we applied it on the logits of the output of the network $log(f(x)) - log(1 - f(x))$.

the supervised data. As Yoo & Kweon (2019) claim that $\mathcal{L}(f(x), y)$ is the actual upper bound of their method, for simplicity we employed the same $\mathcal{L}$ to select uncertain samples. However, as we have seen in Section 3.1, even in the best case scenario this strategy does not work very well in complex problem as it mostly end up selecting outliers and making network convergence more difficult. Both **KCENTER** and **KMEANS**, instead, base their selection criteria only on distance metric on the input data distribution [8]. **KCENTER** aims at covering as much as possible the input data distribution, by selecting at each iteration the furthest sample to the current set of labelled samples (Z_{CENTER}). On the contrary, **KMEANS** strategy selects the closest unlabelled sample to the set of centroids (Z_{MEANS}), following a curriculum-learning strategy.

A.4 Ablation study: selecting diverse constraint violations and employing an uncertainty-like rule improves the performances

In this section, we analyse the role of selecting samples violating a diverse set of rules and of employing the uncertainty-like rule introduced in Sec. 3.2. We recall that diversity sample selection is achieved by requiring a maximum of $r = p/2$ samples violating a certain rule k when selecting a new batch of p samples to be labelled. The uncertainty-like rule, instead, consists in requiring each predicate $\mathbf{f_i}$ to be either true or false $\bigwedge_i \mathbf{f_i} \oplus \neg \mathbf{f_i}$. This way, in case many predicates for a certain sample have a not well-defined value (e.g., $f_i(x) = 0.5$), the violation of the constraint associated to this rule will be high.

We studied three different scenarios, employing the uncertainty-like rule (Unc), requiring a set of sample violating different rules (Div) or both of them (Div Unc). We compared them with the plain versions both in the case of estimating predictions with a standard classifier (KAL) and when using Monte Carlo dropout (KAL$_D$), leading to a total set of eight configurations. We report the results of these ablation study in Table 12. The results reported as KAL in Section 3.1 are, actually, the version with both features (KAL Div Unc). Indeed, KAL Div Unc and KAL$_D$ Div Unc respectively increase the AUBC by 2.3% and 1.9% on the Animals dataset over KAL and KAL$_D$; on the *XOR-like* and on the *CUB200* dataset, the increase is as well important, although smaller than in the latter case; only in the *IRIS* case, the performances get reduced when requiring samples violating a diverse set of rules. This can be most likely explained by considering the size of the Iris dataset and the fact that we employed only 3 rules, specific for each class. In this setting, it may be more convenient for the network to gather samples related to a specific class (i.e., violating a specific rule) only. Indeed, labelling points related to a distribution already covered (i.e., where the knowledge is mostly respected) with very few examples available can slightly decrease the overall performances.

Strategy	XOR	Iris	Animals	CUB200
KAL	97.38 $\pm$0.40	**94.01** $\pm$4.06	53.86 $\pm$1.79	51.94 $\pm$0.43
KAL Unc	97.46 $\pm$0.40	93.97 $\pm$3.61	54.12 $\pm$0.97	51.98 $\pm$0.35
KAL Div	97.62 $\pm$0.79	93.44 $\pm$4.50	54.56 $\pm$1.09	51.97 $\pm$0.39
KAL Div Unc	97.73 $\pm$0.83	93.60 $\pm$3.87	**56.15** $\pm$1.04	51.98 $\pm$0.35
KAL$_D$	97.45 $\pm$0.49	**94.11** $\pm$4.23	53.67 $\pm$1.64	51.93 $\pm$0.47
KAL$_D$ Div	**97.96** $\pm$0.79	93.32 $\pm$4.14	53.74 $\pm$1.42	51.90 $\pm$0.44
KAL$_D$ Unc	97.57 $\pm$0.51	93.51 $\pm$3.80	53.92 $\pm$1.28	**52.02** $\pm$0.33
KAL$_D$ Div Unc	**97.96** $\pm$0.79	93.32 $\pm$4.13	**55.56** $\pm$1.21	**52.10** $\pm$0.24

Table 12: Ablation study on diverse rule selection (**Div**) and on the uncertainty-like rule (**Unc**), both on the standard KAL and on Monte Carlo dropout version (KAL$_D$). In bold, the two best results. Notice how, the latter increase the AUBC by 2.3% and 1.9% respectively over KAL and KAL$_D$, on the Animals dataset.

[8] In the image recognition tasks, we employed the latent distribution extracted by the convolutional features, as commonly done (Ren et al., 2021; Zhan et al., 2022).

A.5 TRAINING EVOLUTIONS ON THE *XOR-like* PROBLEM

In Figure 5 we report further snapshots of the active selection process on the *XOR-like* dataset. They depict the model predictions similarly to Figure 3, but at different iterations and for all the compared methods. In this figure, it is even more clear that *no* uncertainty-based strategy is capable of discovering novel data distributions. More precisely, nor Entropy, nor Margin, nor LeastConf, nor BALD, nor Adv$_{BIM}$, nor Adv$_{DEEPFOOL}$ can cover the data distribution in the right-bottom angle (for which no samples have been drawn during the initial random sampling) even after 20 iterations. Diversity-based strategy, instead, can cover all data distribution; however, both KCENTER and even more KMEANS cannot perfectly predict samples along the decision boundaries. Finally, it is interesting to notice how the sampling selection performed by the SupLoss resemble the one made by KAL. However, even in the simplified upper case considered in this comparison, the selection process performed by SupLoss drives the network more slowly to convergences with respect to KAL. Indeed, as it can be noticed in the right-most plots, after 20th iterations the selection process in SupLoss has not covered very well yet the centre, differently from the selection process of KAL, which correctly covered this important zone.

A.6 COMPARISON OF THE KNOWLEDGE VIOLATION

As introduced in Section 3.4, we tested whether the proposed method allows the model to learn to respect the knowledge provided by domain experts. In Table 13 we reported an extended version of Table 3, computing the increased percentage of violation of the $\mathcal{K}_{CUB-S}$ knowledge by models trained following the compared active strategies w.r.t. a model trained following the proposed strategy (KAL$_{SMALL}$). All the compared methods (including KALs when equipped with all the knowledge on CUB) violate the $\mathcal{K}_{CUB-S}$ knowledge significantly more (4-13 times) than (KAL$_{SMALL}$).

Table 13: Violation of the $\mathcal{K}_{CUB-S}$ knowledge computed as the increased percentage over the violation of a model trained to respect this knowledge (KAL$_{small}$). The proposed method ensures domain experts that their knowledge is respected by the model on test data significantly more than using standard techniques.

Strategy	Increased Violation
KAL$_{SMALL}$	0.00 ± 38.09 %
ADV$_{BIM}$	591.03 ± 532.47 %
BALD	720.25 ± 251.86 %
Entropy	861.74 ± 519.35 %
Entropy$_D$	812.35 ± 359.23 %
KAL	863.53 ± 226.81 %
KAL$_D$	890.98 ± 299.79 %
KCENTER	530.13 ± 188.74 %
KMEANS	555.23 ± 327.80 %
LeastConf	1334.50 ± 697.47 %
LeastConf$_D$	934.46 ± 505.21 %
Margin	846.05 ± 341.52 %
Margin$_D$	637.26 ± 165.97 %
Random	483.10 ± 285.08 %
SupLoss	804.38 ± 76.31 %

A.7 EXTRACTING THE KNOWLEDGE WITH AN XAI METHOD

Explainable AI techniques are more and more used in literature to mitigate the intrinsic opacity of deep neural networks. In Sec. 3.5 we employed Guidotti et al. (2018) to explain the f model at each iteration, and use the explanations as the base knowledge ($\mathcal{K}_{XAI}$) of the proposed method (KAL_{XAI}). This is a viable solution to employ when no other knowledge is available. More precisely, we trained a decision tree h on the supervised training input data X_s to mimic the behaviour of the network f, i.e., we minimized the following loss $\mathcal{L}_{X_s}(h(x), f(x))$. From the decision tree, we extracted global explanations of each class in the form of, e.g., $\forall x$ Setosa $\Leftrightarrow \neg$ Long_Petal. In computer vision tasks, however, decision trees are not suitable to be employed directly on the raw

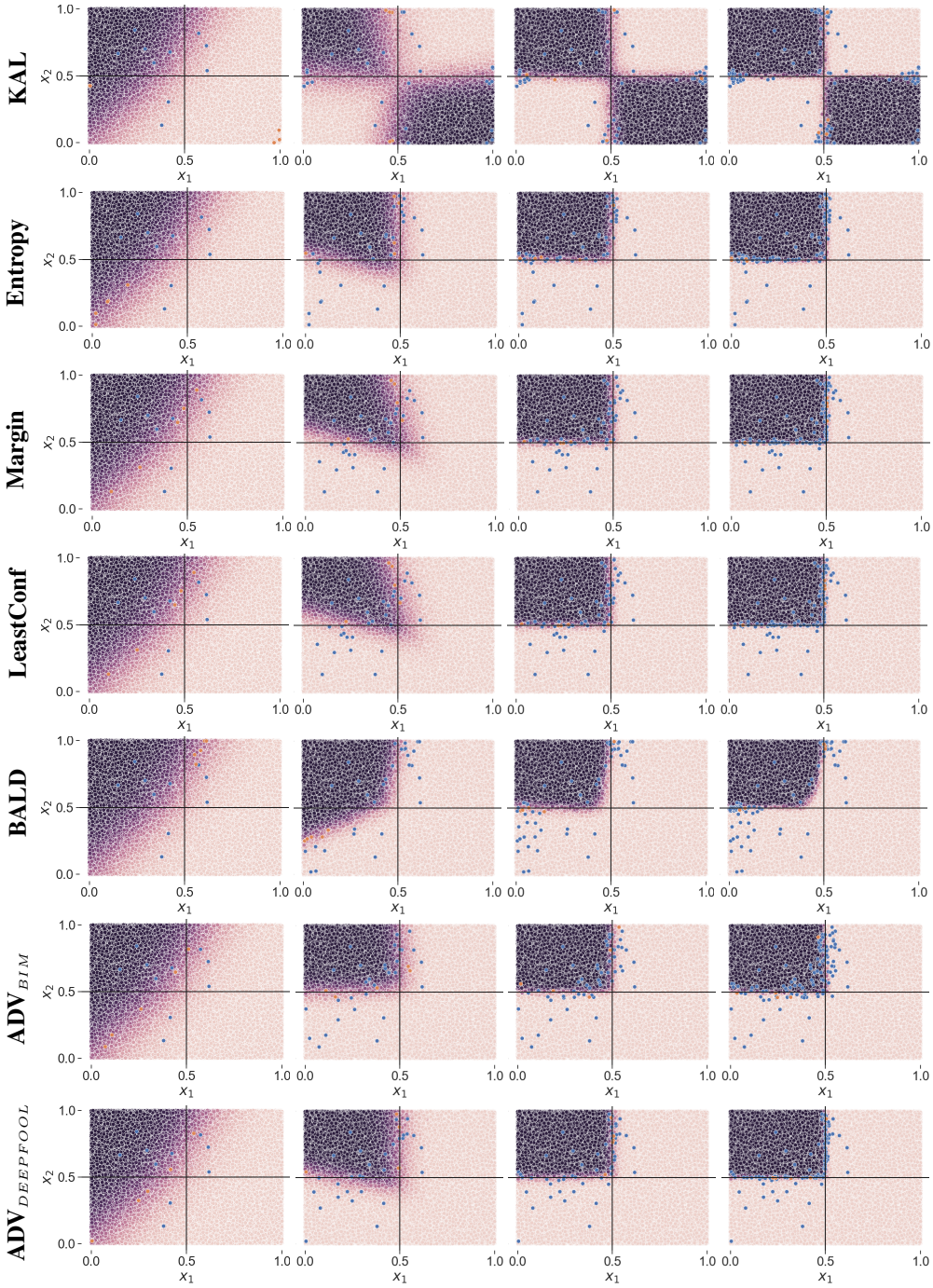

Figure 5: An illustration of the selection process evolution for all the compared strategies on the *XOR-like* problem. We depict network predictions with different colour degrees (light colours negative predictions, dark colours positive prediction); in blue, samples selected in previous iterations, in orange those selected at the current iteration. Black lines at $x_1 = 0.5$ and $x_2 = 0.5$ are reported for visualization purposes only. From left to right, the state at the 1st, 5th, 10th and 20th iteration.

input data. For this reason, following Barbiero et al. (2022), we trained the decision tree to mimic the behaviour of the model f over the main classes when receiving in inputs the attribute ones. The extracted rules, in this case, are of the type Black_footed_Albatross $\Rightarrow$ has_bill_shape_all-purpose $\wedge$ has_underparts_color_yellow $\wedge \ldots$. To also explain the attribute classes, we did the reverse, i.e. we trained the decision tree to mimic the behaviour of the model f over the attribute classes and

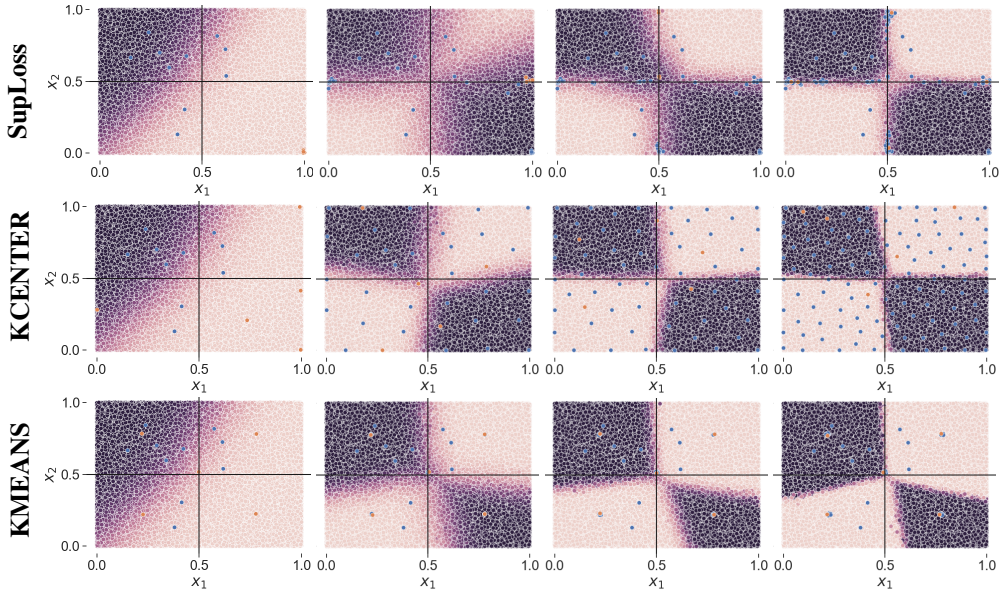

Figure 5: A visual example on the *XOR-like* problem, showing how the training evolves in each of the compared strategy (continued from previous page).

Dataset	# Points	Random	SupLoss	KAL
Dog vs Person	1000-3000	51.27 $\pm$1.41	55.30 $\pm$0.54	**55.90** $\pm$0.39

Table 14: Comparison of the methods in terms of the test mAP (%) AUBC (Zhan et al., 2021) when increasing the number of labelled points on the object recognition task.

receiving in input the main ones. The extracted rules in this case are of the type has_bill_shape_cone $\Rightarrow$ Parakeet_Auklet $\wedge$ Indigo_Bunting $\wedge$ . . . . Finally, we always added to this set of rule a mutual exclusion rule over the main classes and a disjunction over the attributes (where available). Indeed, this notion must be always available as it conditions the choice of the training loss (e.g. binary vs standard cross entropy).

A.8 Employing active strategies in object recognition task

As introduced in Section 3.6, we tested the proposed method also in an object recognition context. As a proof of concept, we experimented on the simple DOGvsPERSON dataset, described in Appendix A.1. In Table 14, we report again the AUBC of the mean Average Precision curves when increasing the budget of labelled points. The same curves are reported in Figure 6. The results in this case are averaged over three different seed initialization of the network. The network increases its more performances when equipped with the KAL strategy (55.90) with respect to standard random sampling (51.41) but also compared to the SupLoss method (55.30). Interestingly, by considering Figure 6, we can appreciate how also in this context the proposed strategy significantly improves the performance of the network already after the very first iterations. Finally, we highlight again that the SupLoss performance reported are an upper bound of the performance of the method proposed in (Yoo & Kweon, 2019). Particularly in this context, we believe that the object recognition supervision loss might not be easily learnt by the external model g. In this proof-of-concept, we only compared with Random and our implementation of the SupLoss method, since they were the most straightforwardly adaptable method to the object recognition context. In future work, we plan to compare with the adapted uncertainty-based strategy proposed in Haussmann et al. (2020).

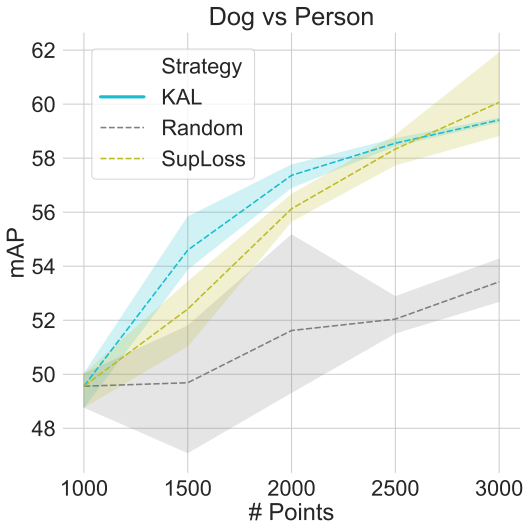

Figure 6: Average test mAP performance growth on the object recognition task when increasing the number of labelled samples. Shadowed areas indicate the 95 % confidence interval.

A.9 Experimental time comparison

In Table 15, we report the complete comparison of the time required by the different strategies. More precisely, we reported the increased percentage in computational time. We included both the time required to select the data and the time required to train the network. The KAL strategy does not come at the cost of a significantly increased time (with an average +0.3-23.4 %), comparably with standard uncertainty based strategies, Entropy (+0.0-32.9%), LeastConf (+0.0-2.01%), and Margin (+0.6-1.8%). Only the Monte Carlo Dropout versions, require higher computational time (KAL$_D$ +5.4-88.3%, Entropy$_D$ +0.0-89.2%, LeastConf$_D$ +0.3-36.6%, Margin$_D$ +1.7-84.0%) since they repeat several times a prediction with different switched-off neurons, to better assess uncertainty. On the contrary, BALD (+44.5-191.4%) and, more importantly, KMeans (+8.3-742.5%), KCenter (+2.0-5446.3%), ADV$_{BIM}$ (+162.4-1353.6%), and ADV$_{DEEPFOOL}$ (+206.8-NA) strategies demand remarkable computational resources, strongly reducing the usability of the same methods. In particular, ADV$_{DEEPFOOL}$ requires a huge amount of resource, since the DEEPFOOL attack is linearly dependent in both the selected samples and the predicted classes. For this reason, it was computationally infeasible to experimented it on the Animals and CUB datasets (respectively with 7 and 200 classes). At last, we reported SupLoss with a *, since the computational time required to compute the cross-entropy loss between the labels and the prediction (as simplified in this comparison), may be significantly different from the one required to predict the loss through the model g.

Table 15: Computational time required to select the samples to annotate by all the compared methods as percentage increase w.r.t the time required for a random sampling as defined in (Zhan et al., 2022). The lower, the better. NA indicates strategies computationally too expensive to compute on certain datasets. Notice how the proposed method is not computationally expensive, contrarily many recent active learning methods proposed in the literature.

Strategy	XOR	IRIS	Animals	CUB
ADV_{BIM}	$+940.38\%$ ± 278.11	$+162.41\%$ ± 31.15	$+1251.36\%$ ± 39.03	$+1353.64\%$ ± 33.53
$\text{ADV}_{DEEPFOOL}$	$+2450.66\%$ ± 1435.04	$+206.79\%$ ± 110.24	NA	NA
BALD	$+191.44\%$ ± 15.14	$+72.08\%$ ± 47.00	$+44.48\%$ ± 3.34	$+112.47\%$ ± 13.28
KCENTER	$+321.06\%$ ± 48.54	$+2.01\%$ ± 1.92	$+606.94\%$ ± 56.57	$+5446.31\%$ ± 59.38
KMEANS	$+8.30\%$ ± 12.39	$+9.39\%$ ± 3.16	$+173.20\%$ ± 9.80	$+742.47\%$ ± 54.30
Entropy	$+3.51\%$ ± 15.95	$+0.04\%$ ± 3.14	$+2.85\%$ ± 3.28	$+32.93\%$ ± 2.67
Entropy_D	$+89.20\%$ ± 14.58	$+0.08\%$ ± 4.24	$+2.98\%$ ± 2.34	$+37.65\%$ ± 2.81
LeastConf	$+0.77\%$ ± 14.70	$+2.01\%$ ± 3.99	$+0.06\%$ ± 0.56	$+0.68\%$ ± 0.07
LeastConf_D	$+83.16\%$ ± 14.59	$+0.31\%$ ± 4.38	$+2.73\%$ ± 3.08	$+36.59\%$ ± 2.72
Margin	$+0.66\%$ ± 14.53	$+1.85\%$ ± 2.56	$+1.26\%$ ± 2.39	$+0.65\%$ ± 0.19
Margin_D	$+84.00\%$ ± 13.94	$+1.69\%$ ± 3.90	$+3.95\%$ ± 3.93	$+38.88\%$ ± 2.49
SupLoss*	$+3.04\%$ ± 14.99	$+1.54\%$ ± 6.26	$+1.61\%$ ± 1.84	$+1.36\%$ ± 1.34
KAL	$+2.52\%$ ± 13.69	$+0.32\%$ ± 2.56	$+6.43\%$ ± 2.02	$+23.42\%$ ± 0.21
KAL_D	$+88.28\%$ ± 20.79	$+5.41\%$ ± 6.23	$+8.18\%$ ± 1.29	$+62.98\%$ ± 2.37

B SOFTWARE

```python
# Knowledge-drive Active Learning - Experiment on the XOR problem
tot_points = 10000
first_points = 10
n_points = 5
n_iterations = 198
seeds = range(5)
x = np.random.uniform(size=(tot_points, 2))
y = ((x[:, 0] > 0.5) & (x[:, 1] < 0.5)) |
    ((x[:, 1] > 0.5) & (x[:, 0] < 0.5))
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.1)


# Defining constraints as product t-norm of the FOL rule expressing the XOR
def calculate_constraint_loss(x_continue, f):
    # discrete_x = (x_continue > 0.5).float()
    discrete_x = steep_sigmoid(x_continue).float()
    x1 = discrete_x[:, 0]
    x2 = discrete_x[:, 1]
    cons_loss1 = f * ((1 - (x1 * (1 - x2))) * (1 - (x2 * (1 - x1))))
    cons_loss2 = (1 - f) * (1 - (1 - (x1 * (1 - x2)) * (1 - (x2 * (1 - x1)))))
    return cons_loss1 + cons_loss2

# Constrained Active learning strategy
# We take the p elements that most violates the constraints and are among available idx
def cal_selection(labelled_idx, c_loss, n_p):
    c_loss[torch.as_tensor(labelled_idx)] = -1
    cal_idx = torch.argsort(c_loss, descending=True).tolist()[:n_p]
    return cal_idx

net = MLP(2, 100)
accuracies = []
used_idx = randint(0, x_train.shape[0], first_points).tolist()
available_idx = [*range(tot_points)]
for n in range(n_iterations):
    train_loop(net, x_train, y_train, used_idx)

    with torch.no_grad():
        preds_train = net(x_train).squeeze()
        preds_test = net(x_test).squeeze()
    accuracy = accuracy_score(preds_test, y_test)
    cons_loss = calculate_constraint_loss(x_train, preds_train)

    available_idx = list(set(available_idx) - set(used_idx))
    active_idx = cal_selection(used_idx, cons_loss, n_points)
    used_idx += active_idx

```

Listing 1: KAL code - Example on the XOR problem. For simplicity we do not consider in this case a train and a test set.

The Python code and the scripts used for the experiments, including full documentation, is freely available under Apache 2.0 Public Licence in a GitHub repository, and it is also provided in the supplementary material. The proposed approach only requires a few lines of code to train a model following the KAL strategy, as we sketch in the code example reported in Listing 1.