Georg Krempl Vincent Lemaire Daniel Kottke
Andreas Holzinger Barbara Hammer

IAL@ECML PKDD 2021

Workshop on Interactive Adaptive Learning

Proceedings

The European Conference on Machine Learning and
Principles and Practice of Knowledge Discovery in Databases
(ECML PKDD 2021)

Bilbao, Spain, September 13, 2021 (virtual conference)

Preface

Science, technology, and commerce increasingly recognise the importance of machine learning approaches for data-intensive, evidence-based decision making. This is accompanied by increasing numbers of machine learning applications and volumes of data. Nevertheless, the capacities of processing systems or human supervisors or domain experts remain limited in real-world applications. Furthermore, many applications require fast reaction to new situations, which means that first predictive models need to be available even if little data is yet available. Therefore approaches are needed that optimise the whole learning process, including the interaction with human supervisors, processing systems, and data of various kind and at different timings: techniques for estimating the impact of additional resources (e.g. data) on the learning progress; techniques for the active selection of the information processed or queried; techniques for reusing knowledge across time, domains, or tasks, by identifying similarities and adaptation to changes between them; techniques for making use of different types of information, such as labeled or unlabeled data, constraints or domain knowledge. Such techniques are studied for example in the fields of adaptive, active, semi-supervised, and transfer learning. However, this is mostly done in separate lines of research, while combinations thereof in interactive and adaptive machine learning systems that are capable of operating under various constraints, and thereby address the immanent real-world challenges of volume, velocity and variability of data and data mining systems, are rarely reported. Therefore, this workshop aims to bring together researchers and practitioners from these different areas, and to stimulate research in interactive and adaptive machine learning systems as a whole. It continues a successful series of events at ECML PKDD 2017 in Skopje (Workshop and Tutorial), IJCNN 2018 in Rio (Tutorial), ECML PKDD 2018 in Dublin (Workshop), ECML PKDD 2019 in Würzburg (Workshop and Tutorial), and virtual ECML PKDD 2020 (Workshop).
The workshop aims at discussing techniques and approaches for optimising the whole learning process, including the interaction with human supervisors, processing systems, and includes adaptive, active, semi-supervised, and transfer learning techniques, and combinations thereof in interactive and adaptive machine learning systems. Our objective is to bridge the communities researching and developing these techniques and systems in machine learning and data mining. Therefore, we welcome contributions that present a novel problem setting, propose a novel approach, or report experience with the practical deployment of such a system and raise unsolved questions to the research community.

All in all, we accepted 10 papers (13 papers submitted) to be published in these workshop proceedings. The authors discuss approaches, identify challenges and gaps between active learning research and meaningful applications, as well as define new application-relevant research directions. We thank the authors for their submissions and the program committee for their hard work.

September 2021

Georg Krempl, Vincent Lemaire, Daniel Kottke
Andreas Holzinger, Barbara Hammer

Organization

Organizing Committee

Adrian Calma	
Andreas Holzinger	Graz University of Technology
Daniel Kottke	University of Kassel
Georg Krempl	Utrecht University
Vincent Lemaire	Orange Labs France

Program Committee

Michael Beigl	TECO, KIT
Mirko Bunse	Dortmund University
Klemens Böhm	Karlsruhe Institute of Technology
Hudelot Céline	Ecole Centrale Paris
Gregory Ditzler	University of Arizona
Michael Granitzer	University of Passau
Marek Herde	University of Kassel
Martin Holena	Institute of Computer Science
Denis Huseljic	University of Kassel
Edwin Lughofer	Johannes Kepler University Linz
Bernhard Pfahringer	University of Waikato
Stefano Teso	Katholieke Universiteit Leuven
Indre Zliobaite	University of Helsinki

Table of Contents

MetaREVEAL: RL-based Meta-learning from Learning Curves *

Manh Hung Nguyen[1,2], Nathan Grinsztajn[3], Isabelle Guyon[2,4], and Lisheng Sun-Hosoya[4]

[1] CentraleSupélec, France, `manh.nguyen@inria.fr`
[2] LISN/Inria/CNRS, Université Paris-Saclay, France
[3] Inria, Univ. Lille, CNRS, France, `nathan.grinsztajn@inria.fr`
[4] ChaLearn, California, USA, {`guyon,sun-hosoya`}`@chalearn.org`

Abstract. This paper addresses a cornerstone of Automated Machine Learning: the problem of *rapidly* uncovering which machine learning algorithm performs best on a new dataset. Our approach leverages performances of such algorithms on datasets to which they have been previously exposed, i.e., implementing a form of *meta-learning.* More specifically, the problem is cast as a REVEAL Reinforcement Learning (RL) game: the meta-learning problem is wrapped into a RL environment in which an agent can start, pause, or resume training various machine learning algorithms to progressively "reveal" their learning curves. The learned policy is then applied to quickly uncover the best algorithm on a new dataset. While other similar approaches, such as Freeze-Thaw, were proposed in the past, using Bayesian optimization, our methodology is, to the best of our knowledge, the first that trains a RL agent to do this task on previous datasets. Using real and artificial data, we show that our new RL-based meta-learning paradigm outperforms Free-Thaw and other baseline methods, with respect to the Area under the Learning curve metric, a form of evaluation of *Any-time learning* (i.e., the capability of interrupting the algorithm at any time while obtaining good performance).

Keywords: Meta-Learning · Learning Curves · Reinforcement Learning.

1 Introduction and related work

Meta-learning in machine learning refers to learning from prior experience on other datasets than the current dataset of interest. There are many meta-learning settings, including learning from **Model Evaluations**, learning from **Task Properties**, and learning from **Prior Models** [31]. In this paper, we address a particular setting of meta-learning in which the goal is to **rapidly** find an algorithm that performs best on a new dataset. Since speed is of the essence, rather than fully training all algorithms, we interrupt (then eventually resume) training. Hence, we allow our meta-algorithm to switch between learning curves.

* The first author contributed most, the others are in alphabetical order of last name. Supported by ANR Chair of Artificial Intelligence HUMANIA ANR-19-CHIA-00222.

Our setting belongs to the family of meta-learning **Model Evaluations** methods, which make use of pre-defined performance measures, e.g. test accuracy and training time. One baseline approach is to select the algorithm performing best on previous datasets, e.g. according to average rank [1, 17]. Other prior art approaches include recommender systems for Meta-learning [7, 22, 23, 27, 29, 35], largely dominated by Collaborative Filtering methods (e.g. Matrix Factorization). In this line of work, ActivMetal [29] has inspired our approach. Our work is mostly in line with [28], casting the problem as a REVEAL game, a subclass of Markov Decision Processes.

Task Properties (meta-features) describe the characteristics of datasets. They may include statistical information, information-theoretic measures, or learned meta-features. In Meta-Regression, regression algorithms are used to predict the performances of algorithms based on the meta-features of the problems (and meta-features of the algorithms). One could estimate a classifier's performance by exploiting relationships between the dataset properties and the classifier's performance [3]. Kopf et al. explored deeper the choices of measurements for dataset characterization [11]. Another work made by Guerra et al. used Support Vector Machines to predict the Performance of Learning Algorithms [8]. In general, meta-regression highly depends on the quality of the meta-features used. Our present approach is not a *Task property* method, since it does not rely on such meta-features, although they could be added in the future.

Learning from **Prior Models** usually focuses on transfer learning and few-shot learning applied to deep learning models. While the former uses models trained on source tasks as starting points to develop models for a new target task, the latter aims at training a good model given very few training examples. Much progress has been made in these settings with some state-of-the-art methods, such as MAML [6], Reptile [24], MetaOptNet [12], and R2-D2 [4]. Our method does not leverage *prior models*, although this could be done in future work.

The setting considered in this paper is **active meta-learning**, where an agent actively requests to train and test algorithms to reveal their performance on a given dataset. We fuse three ideas: (1) that of "active meta-learning" exploited in ActivMetal [29], that of using Reinforcement Learning exploited in [28] by framing the meta-learning problem as a REVEAL game, and that of learning from partial learning curve information used in Freeze-Thaw, proposed for hyper-parameters optimization and model selection [30] (without any meta-learning).

Compared to previous approaches, we gain in speed and accuracy: Both ActivMetal and REVEAL are computationally demanding since they require fully training and evaluating models. Our new method using partially trained models (along the learning curve) is thus more effective. Furthermore, ActivMetal requires multiple computationally expensive matrix factorizations using the entire meta-dataset of past scores. Our method based on pre-trained policies does not require storing and using past scores on other datasets at utilization time. Finally, Freeze-Thaw, which inspired us to use learning curves, relies on heuristic policies derived from human expertise, not trainable agents performing meta-learning, which is the setting considered in this paper. Other learning-curve based methods [13–15] rely on pairwise comparisons of algorithms, which would not scale well with the number of algorithms and involve"hard-coded" policies (no meta-learning). Our principal contributions are:

1. We introduce **meta-learning environments** using learning curve information with two reward functions specifically designed for Fixed-time learning and Any-time learning. These two types of learning are described in Section 3.1 and Section 3.2.
2. We implement and evaluate various **RL agents and baseline methods** on a meta-dataset from the AutoDL challenge [19] and a novel artificial meta-dataset. We experimentally show that RL agents can "meta-learn" the underlying structure of training meta-datasets to later solve similar learning tasks more efficiently.
3. We propose a **Switching Frequency (SF) metric** to quantify how often an agent pauses running an algorithm and switches to running another one during an episode. This metric is related to the trade-offs between exploitation and exploration. We study the correlation between this metric and the cumulative reward achieved by the agents.

2 Mathematical statement of the problem

2.1 Meta-learning as algorithm recommendation

Meta-learning is learning to learn. In this paper, we consider the algorithm recommendation setting of meta-learning: The goal is to find, from a set of algorithms, the algorithm performing best on a new dataset, given the experience of these algorithms on previous datasets. This experience can be embedded in a meta-dataset.

Definition 1. *(Meta-dataset). A meta-dataset of m algorithms on n datasets can be expressed as a performance matrix P with a size of $(m \times n)$, where column j (for $j = 1, ..., n$) corresponds to algorithm A_j, row i (for $i = 1, ..., m$) corresponds to dataset D_i, and $P(i, j)$ is the performance score of A_j tested on D_i.*

Definition 2. *(1D Meta-Learning Problem). Given a meta-dataset P with a size of $((m-1) \times n)$, a new dataset D_m, and the partial performance information I_m of algorithms on this new dataset D_m (which is progressively revealed at a given cost), the meta-learning problem is to find the best algorithm A_{j^*} for D_m such that:*

$$j^* = \underset{j=1,\ldots,n}{\operatorname{argmax}} P(m, j) \tag{1}$$

From *Definition 1*, we concentrate on **zero-level** meta-learning, as defined in [18]. Meta-learning algorithms are categorized in 3 families, related to the taxonomy of [31] into *Model Evaluation*, *Task Properties*, and *Prior Models*, but based on the *level of information used*:

– **Zero-level meta-learning,** or black-box meta-learning: Only past performances of *Model Evaluations* (e.g., accuracy score on datasets).
– **First-level meta-learning,** or gray-box meta-learning: Performance scores, dataset meta-features (i.e. *Task Properties*) and/or algorithm hyper-parameters.
– **Second-level meta-learning,** or white-box meta-learning: First and second level information is complemented by full knowledge of the datasets and inner functioning of the algorithms (related to the notion of *Prior Models*).

From *Definition 2*, we concentrate on **1D meta-learning**. Meta-learning was divided into 1D meta-learning and 2D meta-learning in [28]. In 1D meta-learning, a search for the best algorithms for a single dataset at a time is performed. In 2D meta-learning, good matches of algorithm-dataset pairs $\{D_i, A_j\}$ are seeked over the 2D score matrix (initialized with many missing values).

2.2 REVEAL games

In this section, we relate meta-learning problems to REVEAL games, which has been previously discussed in [28]. Meta-learning problems, in the recommendation setting introduced in the previous section, can be cast as REVEAL games, a particular class of Markov Decision Processes (MDP), amenable to Reinforcement Learning [28]. Since we will be using this framework, we first briefly recall what REVEAL games are.

In a REVEAL game, the agent's action can only influence the amount of information it can gain, not the underlying data generative process, i.e., the agent's actions have no influence over the course of the "world". Consequently, a good operational test of whether a MDP is a REVEAL game is to find out whether it is possible to pre-compute all states and rewards *a priori*, before the start of a game episode. A simple metaphor for a REVEAL game is a "game board" covered with "cards". Each card is associated with some information. When the game starts, all cards are placed face down, such that the information is hidden from game players or agents. The goal of an agent is to move around the board and reveal the card's information to maximize rewards received in an episode. Examples of REVEAL games include Battleship [32], Mouse in a maze [2], Minesweeper [33], Pacman [34] without ghosts, etc. One example of a game that is not a REVEAL game is the Pacman but with ghosts because the agent's moves affect the motions of the ghosts.

Meta-learning problems can be viewed as REVEAL games where a new dataset corresponds to a new board. An action of the agent on the board is a choice of pair $\{training\ algorithm, dataset\}$ yielding a reward based on the performance achieved by the chosen algorithm on the chosen dataset. Figure 1 shows an overview of how a meta-learning problem is related to a REVEAL game.

As an additional twist, "cards" in REVEAL games can be partially or progressively revealed. This metaphor portrays well the case in which learning machines are progressively trained, and revealing a card step-by-step corresponds to obtaining the next performance of the algorithm after training one more epoch.

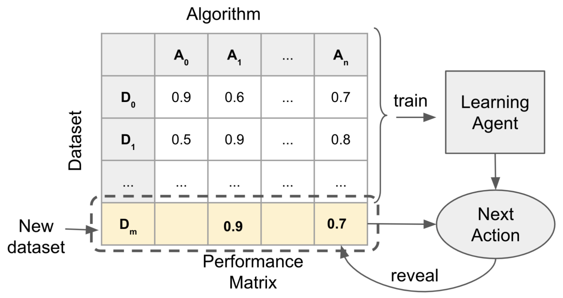


Fig. 1: Meta-learning as a REVEAL game where agents progressively reveal the performance of algorithms on a given new dataset and try to find the best algorithm for that dataset as fast as possible.

Concerning the meta-datasets, this corresponds to adding one more dimension (time or number of epochs) in the meta-dataset performance matrix to store an entire **learning curve** as opposed to a single final score, as explained further in the next section.

3 MetaREVEAL

In this section, we introduce RL-based meta-learning from **learning curves**, which are an essential ingredient for time management in the search for the best performing algorithm. Indeed, training all algorithms fully (to the point of reaching asymptotic training performance) is wasteful, considering that the least promising algorithms can be abandoned early on. Given a limited time budget, it is therefore preferable to probe first the performance of algorithms by training them only a few epochs, then eventually train more certain algorithms, perhaps switching back and forth as more of the learning curve is revealed. The goal of our RL agent is to uncover an optimal strategy, monitoring exploration and exploitation.

We investigate two settings, which have implications in the time management and the exploration-exploitation tradeoff: **Fixed-time Learning** and **Any-time Learning**. In the Fixed-time Learning setting, an overall time budget is given, and the goal of the agent is to find the best algorithm before the time is out. The agent can therefore explore freely within this time budget, without the need to find a good solution early on. In contrast, in the Any-time Learning setting, the agent can be stopped and judged for its performance at any time. There is therefore pressure on the agent that it finds a good solution early on and keeps improving it incrementally. Figure 2 shows a concrete example of two algorithms competing to show the difference between Fixed-time learning and Any-time learning settings. We introduce meta-learning environments designed for each setting.

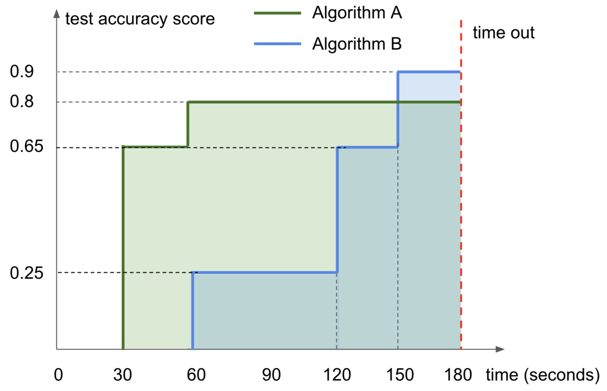


Fig. 2: **Fixed-time learning versus Any-time learning**. In *Fixed-time learning*, within a time budget $\mathcal{T} = 180$ seconds, algorithm B obtained a higher final test accuracy score (0.9) than algorithm A (0.8), making algorithm B the winner in this setting. However, in *Any-time learning*, we use the Area under the Learning Curve (ALC) metric to compare. Thus, algorithm A clearly outperformed algorithm B in this setting. If both algorithms were to stop at any point in time, algorithm A would most likely achieve better performance than algorithm B, indicating that algorithm A possesses a greater capacity for any-time learning.

3.1 Fixed-time Learning

In the Fixed-time Learning scenario, an agent is given a total time budget $\mathcal{T}$ to be spent on training any algorithm in the algorithm set A. The agent's goal is to find the best algorithm for a given dataset within the time budget. $\mathcal{T}$ may be varied to have the agents exposed to different conditions (e.g., $\mathcal{T}$ is drawn uniformly from a pre-defined set of time budgets).

Definition 3. *(State). A state s_t is a matrix of dimensions $2 \times n$, which consists of two channels represented by two vectors with the same length of n. The first vector T stores the amount of time that has been spent so far for each algorithm and the second vector V represents the current test score of each algorithm (current value on the learning curve) in the current episode:*

$$T = [t_j] \;\; for \;\; j = 1, ..., n \tag{2}$$

$$V = [v_j] \;\; for \;\; j = 1, ..., n \tag{3}$$

where n is the number of algorithms. At the beginning of an episode, all values of t_j are initialized to 0 and v_j to -1, to indicate that performances of algorithms A_j have not been revealed yet.

Definition 4. *(Action). An action is to start/continue training an algorithm in a fixed amount of time Δt (pre-defined by the environment creator, e.g., $\Delta t = 10$*

seconds) and then make predictions on the test data to receive the next test score.[5] *For simplicity, we define an action by the corresponding algorithm index:*

$$a_t = j, \tag{4}$$

where j is the index of the algorithm A_j which is going to be trained and tested next. Once the action a_t is done, t_j and v_j in the state are updated to form the next state.

Definition 5. *(Fixed-time Learning Reward Function). A shaping reward function based on performance improvement, which gives rewards more frequently to the agent and lets the agent knows that it is getting better and getting closer to the best algorithm:*

$$r(t) = V^*(t) - V^*(t - \Delta t) \ , \tag{5}$$

where $V^*(t)$ and $V^*(t - \Delta t)$ are the best algorithm performances found in this step and the previous step respectively:

$$V^*(t) = \max_{k \leq t} V(k) \ , \tag{6}$$

$$V^*(0) = 0 \ , \tag{7}$$

Definition 6. *(Termination condition). An episode ends when $\mathcal{T}$ is exhausted.*

At the end of the episode, the **cumulative reward** is equal to $V^*(\mathcal{T}) - V^*(0) = V^*(\mathcal{T})$, **the score of the best algorithm found within the time budget $\mathcal{T}$**. Our agent therefore implements a meta-algorithm whose (meta-)learning curve is given by $V^*(t)$, but it is judged only by its end result.

3.2 Any-time Learning

In the Any-time Learning setting, we want to encourage the agent to obtain a meta-learning curve, which is steep at the beginning, i.e., to uncover **good algorithms as fast as possible**. In this way, even if the agent is stopped early, we will get as good performance as possible, thus obtaining Any-time Learning capabilities.

States, actions, time budgets, and termination conditions are defined similarly as in Fixed-time learning. We designed a specific reward function for this type of learning:

[5] Agents' action are based on the test performance V_j, which is assumed to be accurate and a good approximation of the generalization error (i.e. we assume large test sets and very small error bars). In this work, we focus on meta-learning, hence, the problem of possibly "overfitting/underfitting the test set" is not discussed in this paper and left for future works.

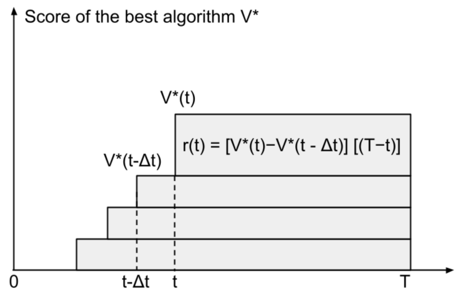

Score of the best algorithm V*
V*(t)
V*(t-Δt)
r(t) = [V*(t)−V*(t - Δt)] [(T−t)]
0
t-Δt
t
T

Fig. 3: Computation of the ALC.

Definition 7. *(Any-time Learning Reward Function). This function puts more emphasis on performance improvement at the beginning of an episode. The reward is defined by:*

$$r(t) = [V^*(t) - V^*(t - \Delta t)]\ [(\mathcal{T} - t)] \tag{8}$$

The weight $[(T - t)]$ is the only difference compared to the reward function in Fixed-time Learning. If we scale the x-axis logarithmically then the reward function becomes:

$$r(t) = [V^*(t) - V^*(t - \Delta t)]\ [(1 - \tilde{t})] \tag{9}$$

with

$$\tilde{t} = \frac{\log(1 + t/t_0)}{\log(1 + \mathcal{T}/t_0)} \tag{10}$$

The larger t_0 is, the more important the beginning of the learning curve is. If $\mathcal{T} \gg t_0$, then $\tilde{t} \to 0$ and the reward function becomes equivalent to that of Fixed-time Learning (Equation 5). In our experiments, t_0 is set to 50 (seconds).

At the end of the episode, the **cumulative reward** will be the **Area under the Learning Curve** (ALC) within the time budget $\mathcal{T}$. The computation of the cumulative reward can be carried out by integrating the learning curve using horizontal rectangles, in the style of Lebesgue integrals (Figure 3). The ALC metric was used in the AutoDL challenge with the same purpose of emphasizing Any-time Learning [20, 21].

4 Experiments and Results

In this section, we first describe how the meta-datasets used in our experiments are obtained. Then, we discuss the experimental results and findings from running the implemented RL agents and baselines on the meta-datasets. The code for reproducing the experiments is available on our Github repository [6].

[6] https://github.com/hungnm2008/metaREVEAL.git

4.1 Meta-datasets

We use learning curves collected from the AutoDL Challenge [20] to build our first meta-dataset. However, since this meta-dataset is quite small and not complete, we generate artificial learning curves using parameterized sigmoid functions. Both of them will be discussed in detail below.

Learning curves from the AutoDL challenge [20]. This meta-dataset is made by the predictions of 13 Automated Deep Learning algorithms on 66 datasets in the AutoDL Challenge [20]. These algorithms include top 9 algorithms and 4 baselines competed in the challenge. The fact that we use a meta-dataset from the AutoDL challenge might cause a misunderstanding that our method is comparable to the methods competed in the challenge. However, we are doing one level up, meta-learning from past performances of these AutoDL methods. The score used in the challenge is the Area under the Learning Curve (ALC) computed using the Normalized Area Under ROC Curve (NAUC) scores gathered during the learning process. The NAUC score is obtained by making predictions on the test set at any timestamp during 20 minutes. One difficulty is that each algorithm in the meta-dataset made predictions at different timestamps while our agents do it regularly every Δt seconds. Thus, some data points on the learning curves at desired timestamps are not available to the agents. In this case, the learning curve's most recent value (data point) will be returned. The learning curves obtained from the AutoDL challenge are not monotonic. During the competition, some algorithms' performances decrease after some time of training.

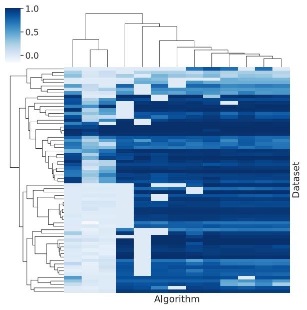
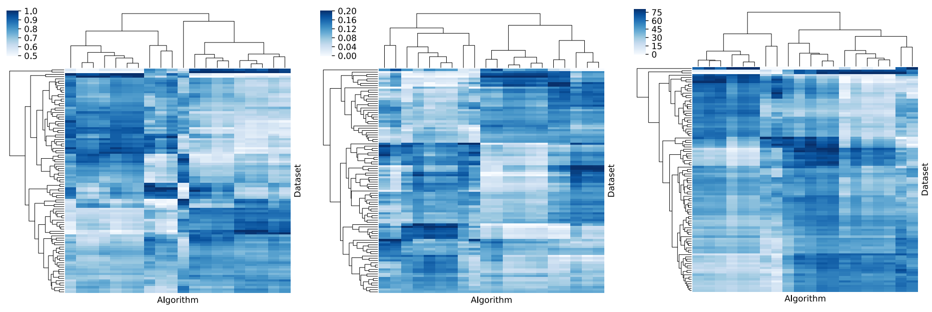


Fig. 4: (AutoDL meta-dataset) Hierarchically-clustered heatmap showing *nauc_mean* score of algorithms on datasets in the AutoDL meta-dataset. The figure demonstrates that there is some structure in the data, which can potentially be exploited by the learning agents. The 'blocks' indicate that some algorithms are more suitable for solving some dataset tasks. This is some transferable knowledge that the agent can learn.

Artificial Learning Curves. We have created an artificial meta-dataset that contains learning curves of 20 algorithms on 100 datasets. The purpose of creating these curves is to have a meta-dataset with a larger size, no missing data, and containing underlying structure indicating some groups of algorithms are good for some groups of datasets. We assume these learning curves have the S-shape-like sigmoid curves, hence, they are monotonically increasing by definition. Each learning curve of algorithm A_j on dataset D_i is a sigmoid function defined by three parameters a, b and c as follows:

$$lc_j^i = \frac{a}{1 + e^{-b*(x-c)}} \tag{11}$$

These parameterized functions allow us to experiment with various learning curves, by adjusting their asymptotic performance (specified by a), increasing rate (specified by b), and "warm-up" time (specified by c). Values of each parameter a, b, and c are shown in matrices in Figure 7. Each matrix was constructed from a matrix factorization, which means it was obtained as a product of three matrices $U\Sigma V$ where U and V are random orthogonal matrices and Σ is a diagonal matrix of "singular values". The values are then scaled to desired range for each parameter.



(a) Parameter a (a) Parameter b (a) Parameter c

Fig. 7: (Artificial meta-dataset) Hierarchically-clustered heatmaps showing values of the three parameters used to build the artificial learning curves. Blocks appear, revealing that some groups of algorithms have correlated parameter values on groups of datasets (e.g. learning curve asymptotic value, controlled by the parameter a). The learning agents are expected to learn such properties and output an effective search strategy.

4.2 Reinforcement Learning Agents

In Reinforcement Learning, the goal of an agent is to find a policy that maximizes expected (discounted) rewards. Reinforcement Learning methods can be categorized into *value-based*, *policy-based*, and *hybrid* methods. Value-based methods learn a value function that is used to evaluate a state or a state-action pair. Then the policy is derived directly from the value function. In contrast, policy-based methods explicitly learn a representation of a policy and keep updating it during learning. Many hybrid approaches learn both value function and a policy simultaneously gain great success in RL. Actor-Critic is a well-known architecture used in these hybrid approaches, where the "Critic" computes estimated values and the "Actor" updates the policy according to the values provided by the Critic. We have chosen a diverse group of RL agents due to their characteristics and their strategies to be evaluated in our experiments:

Double Deep Q Networks (DDQN) [10]: value-based, off-policy, ε-exploration strategy.

Soft Actor-Critic (SAC) [9]: hybrid (actor-critic architecture), off-policy, entropy-based exploration strategy.

Proximal Policy Optimization (PPO) [26]: hybrid (actor-critic architecture), on-policy, entropy-based exploration strategy.

4.3 Baselines

We compare the performance of RL agents with established baseline methods, which allow us to select an algorithm that should perform well on a novel dataset.

Freeze-Thaw Bayesian Optimization. [30]. This method aims at efficiently searching for good model hyper-parameters. It maintains a set of "frozen" models that are partially trained and takes advantage of the partial information to decide which ones to "thaw" and resume training. This avoids spending too much time on bad models, and only promising models should be exploited more. Freeze-Thaw requires hyper-parameters to be able to search for good models. However, we are working on Zero-level meta-learning, hyper-parameters are not considered in choosing an algorithm (model). We made some changes to make the Freeze-Thaw method able to run in our experiments. The performance matrix has been arranged so that similar algorithms are placed together. Then we use the algorithm index as a "hyper-parameter" that describes and represents the locality of the algorithm in the searching space.

Average Rank. Inspired by these works [1, 5, 16, 17], we build a global ranking of algorithms across training datasets. This is done in the training phase by running all algorithms for all training datasets and taking the average of their ranks to form the final ranking. The global average rank for each algorithm A_j is obtained by:

$$global\_rank(A_j) = \frac{\sum_{i=1}^{D_{train}} rank_j^i}{D_{train}} \tag{12}$$

where D is the number of training datasets, and $rank_j^i$ is the rank of algorithm A_j on the dataset D_i. Given a new test dataset, only the algorithm with the highest global rank is selected to run with the entire time budget $\mathcal{T}$. This baseline is very time-consuming in practice since it needs to try all algorithms on all datasets in training.

Best on Samples. This baseline is adapted from [25] by using a fixed amount of time $t_{sampling}$ instead of a fixed number of samples. At the beginning of each episode, it trains each algorithm with the same amount of time $t_{sampling}$ and then selects the one that performed best within $t_{sampling}$ to run with the remaining time budget. In our experiments, we set $t_{sampling} = \Delta t$.

Random. This baseline performs a random search over the algorithm space. Each action is to randomly choose an algorithm for training and testing within Δt. This baseline has a **very large variance**. When we report results, we first average results over 5 trials of the random search method, therefore reducing its variance, and report average performance. One needs to bear in mind though that this is just for comparison purposes and in *not a realistic setting* (because in practice one would not average over several runs, this is impossible because once the performances of algorithms are revealed, one cannot take them back).

4.4 Setup and Evaluation Metrics

We train the agents in two learning scenarios: **Fixed-time Learning** and **Any-time Learning** using two meta-datasets: the **AutoDL meta-dataset** and the **Artificial meta-dataset**. Since these meta-datasets are quite small, we use k-Fold Cross-Validation with $k = 4$ to train and test the agents.[7]

To compare the agents, we use two metrics: **Average Cumulative Reward** and **Average Switching Frequency** (defined in Definition 8). The means of cumulative reward and switching frequency are calculated for each test fold. The final average cumulative reward, average switching frequency, and their corresponding standard deviations are computed over all folds.

Definition 8. *(Switching Frequency). We proposed a Switching Frequency (SF) metric for evaluating how frequently an agent switches between algorithms. In an episode, the SF value of an $agent_k$ is defined as:*

$$SF(agent_k) = \frac{\sum_{t=1}^{\mathcal{T}} \mathbb{1}_{a_t \neq a_{t-\Delta t}}}{\mathcal{T}/\Delta t} \tag{13}$$

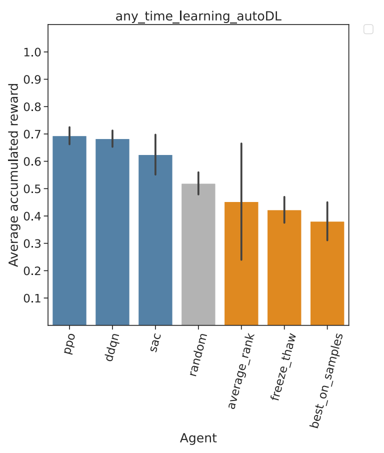
with $\mathcal{T}$ is the total time budget, Δt is the amount of time spent for an algorithm in one step.

4.5 Results

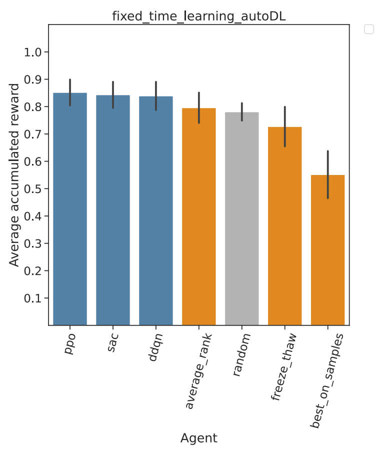
We discuss our experimental results in two learning scenarios and focus on two points: (i) the average cumulative reward and (ii) the correlation between average cumulative reward and average switching frequency.

Any-time learning, (Figure 8a, 8c, 10a, 9c). The results indicate that a good strategy to be successful in Any-time learning is to bet at the beginning on algorithms that performed well on past datasets and stick to them to climb the learning curve fast, then start exploring. This is illustrated by the *ppo* agent, which obtained the highest cumulative reward, followed by other RL agents.

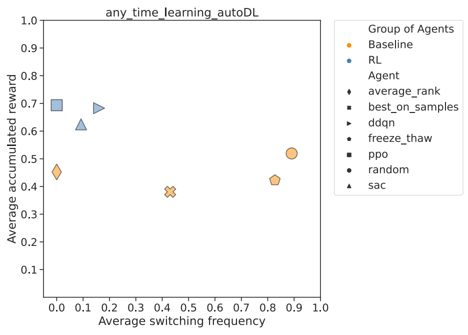
[7] This violates the assumption that we have large test sets made earlier and is a limitation of this mode of evaluation.



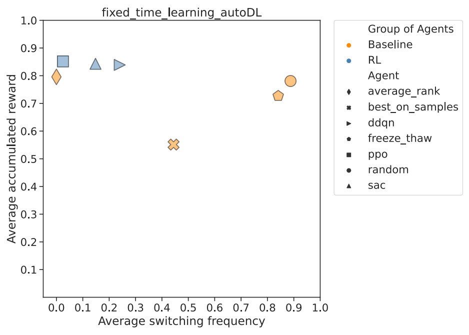
(a) Cumulative reward in *Any-time learning*



(b) Cumulative reward in *Fixed-time learning*

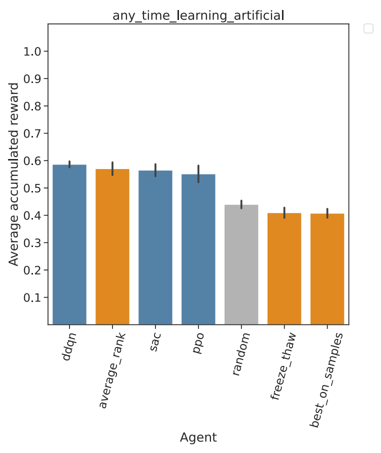


(c) Correlation between Switching frequency and Cumulative reward in *Any-time learning*

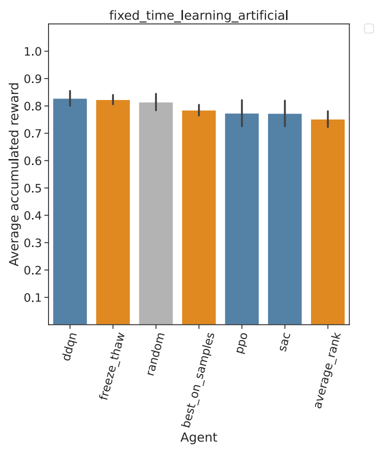


(d) Correlation between Switching frequency and Cumulative reward in *Fixed-time learning*

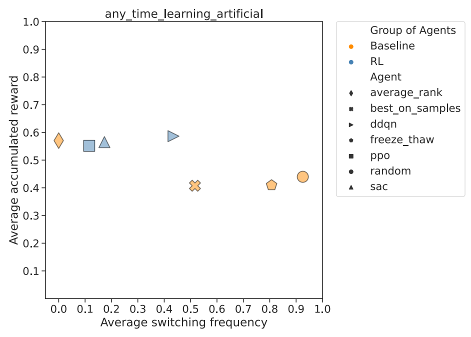
Fig. 8: Experimental results on the **AutoDL meta-dataset**. Time budget $\mathcal{T}$ is drawn uniformly from [200, 300, 400, 500] (seconds) and Δt is set to 10 (seconds). From the bar plots, it can be seen that RL agents (in blue color) outperformed baselines in both learning settings, more significantly in the *Any-time learning*. These RL agents tend to have very low switching frequency, as shown in the scatter plots. In *Any-time learning*, the *average_rank* agent has a high error bar because the algorithm chosen by the agent does not consistently perform well at the beginning of an episode. To stress that the "random" agent is an average over 5 random runs, we highlight its bar in gray. Its total variability is higher than suggests the 4-fold error bar represented in sub-figures (a) and (b).



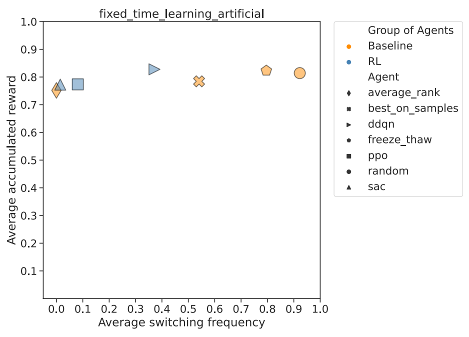
(a) Cumulative reward in *Any-time learning*



(b) Cumulative reward in *Fixed-time learning*



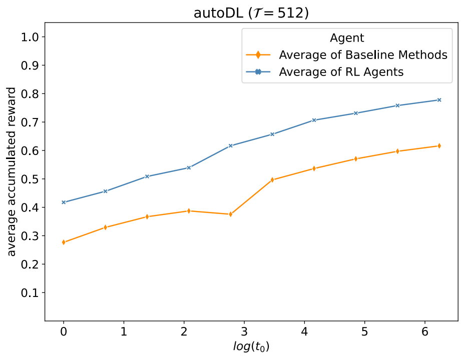
(c) Correlation between Switching frequency and Cumulative reward in *Any-time learning*



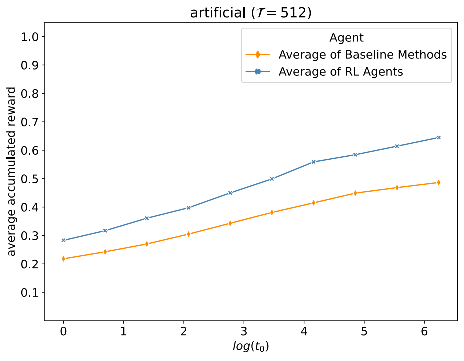
(d) Correlation between Switching frequency and Cumulative reward in *Fixed-time learning*

Fig. 9: Experimental results on the **Artificial meta-dataset**. Time budget $\mathcal{T}$ is drawn uniformly from [500, 700, 900, 1100] (seconds) and Δt is set to 20 (seconds). In *Any-time learning*, RL agents and *average_rank* agent achieved similar cumulative rewards and higher than the rest of the baselines. These agents do not switch algorithms frequently as the others, as shown in the scatter plots. In *Fixed-time learning*, within the given time budget $\mathcal{T}$, all agents performed almost the same.

They are among the algorithms with the lowest switching frequency. Their low switching frequency can explain their success at the beginning of the learning curve, as they favor more exploitation than exploration. In contrast, the policy of *best_on_samples* and *freeze_thaw* forces agents to try each algorithm at least once at the beginning (train and test the algorithm in Δt first seconds). Thus, if they manage to find the best algorithm, this should happen only near the end of the episode, which makes it less valuable in the Any-time learning setting. This explains why they performed worst in the Any-time learning setting in both meta-datasets. We vary the value of t_0 to investigate its influence on agents' performances. More precisely, the value of t_0 is drawn from the set: $[1, 2, 4, 8, 16, 32, 64, 128, 256, 512]$, while the time budget $\mathcal{T}$ is set to 512. The results of this experiment are shown in Figure 10.



(a) AutoDL meta-dataset



(b) Artificial meta-dataset

Fig. 10: Tuning hyperparameter t_0 in Any-time learning. We compare the average accumulated reward of RL agents (in blue) and baseline methods (in orange). The x-axis shows the value of t_0 on a log scale. In Any-time learning, changing t_0 leads to changing the reward function. Thus, the purpose of these figures is not to show that agents achieve higher rewards when t_0 increases. The key finding is that the performance difference between RL agents and baseline methods gets *larger* as t_0 increases, indicating that RL agents can learn better when we emphasize more on the any-time learning capability (with a high value of t_0). The difference is more obvious in the Artificial meta-dataset, which can be explained by the chosen time budget $\mathcal{T}$. In the AutoDL meta-dataset, the time budget $\mathcal{T}$ of 512 is large enough for the baseline methods to maintain the difference with the RL agents when t_0 increases, which is not the case in the Artificial meta-dataset.

Fixed-time learning, (Figure 8b, 8d 10b, 9d). In both meta-datasets, the winner is a RL agent. In the AutoDL meta-datasets, RL agents achieved higher cumulative rewards than the baselines. However, in the Artificial meta-dataset, there was no significant difference between all agents. Within the given time budget $\mathcal{T}$, all agents managed to find a good algorithm at the end. This emphasizes the fact that learned policies to manage time budget are mostly beneficial in the Any-time learning setting, where monitoring the exploration-exploitation tradeoff is critical.

Comparison between datasets. The AutoDL meta-dataset has a clear block structure in the vertical (dataset) direction, which means there is some algorithm ranking transferable across datasets in the same group. The fact that RL agents outperform others in both any-time and fixed-time learning indicates that the RL agents successfully meta-learn those rankings, which let them finds the best algorithms for similar datasets with less exploration than other agents that cannot meta-learn (best on samples, freeze-thaw, random or average-rank that uses the same ranking for all datasets), this make RL agents shine even more in any-time learning. The structure of the artificial dataset is more subtle and harder to learn, as it appears. More work needs to be done to fully elucidate this.

5 Conclusion

Meta-learning can be viewed as a sequential decision-making problem where an agent selects and trains algorithms progressively for a given dataset. The goal is to find the algorithm performing best within a fixed amount of time (Fixed-time learning) or at any time (Any-time learning). We have proposed learning environments that allow RL agents to learn policies (as opposed to hard-coding them) using past experiences on similar datasets (meta-learning). Trained agents operate by training algorithms step by step, thus revealing their learning curves. By doing so, they create a meta-learning curve from the performance of the best algorithm revealed so far.

Both knowledge from past dataset experience (captured in the learned agent policy), and current information on the dataset at hand (embedded in the current state) are used by agents to make decisions. By leveraging partial learning curve information, an agent may stop training algorithms that are not promising and concentrate hardware resources on an algorithm that has more potential to be the best-performing one on the given dataset, which would save a huge amount of time. In both Any-time and Fixed-time learning, the RL agents successfully acquired two important skills: (1) Meta-learning, which allows trained RL agents to identify good algorithms with less exploration for new datasets thanks to the previous training, this is more prominent in Any-time learning; (2) Exploration-exploitation trade-off, which explains the different policies they derive in Fixed-time and Any-time settings. In Any-time learning, RL agents obtained a higher cumulative reward (Area under Learning Curve) than the baselines. In contrast, in Fixed-time Learning, all methods obtain a similar cumulative reward (best final score). From a RL perspective, this outlines that the Any-time learning problem offers more possibilities to learn clever policies monitoring the exploration-exploitation trade-off. When the number of algorithms increases, MetaREVEAL with RL agents would show more advantages over the baselines in terms of computational time (e.g. the average_rank agent needs to try all algorithms on the training datasets). In addition, if we have numerous sets of hyperparameters of the same model, we can adapt MetaREVEAL to work with continuous action spaces, which would be more efficient in searching for the optimal set of hyperparameters.

Future work includes performing more experiments on the artificial data, varying its parameter settings, to elucidate relationships between data structure and policy learning. Work is also under way to apply our method to other real-world meta-datasets. Systematic experiments must be performed to vary values for the parameters of our meta-learning RL environments: $\mathcal{T}$ and t_0. Last but not least, it would be interesting to do some theoretical research and propose RL methods more dedicated to the meta-learning REVEAL game setting and investigate the computational complexity of such methods. We would also like to extend this work to the First-level meta-learning, Second-level meta-learning, and 2D meta-learning problems.

A Appendix A - Full Experimental Results

Table 1: **Average Cumulative Reward** achieved by each agent in each setting. Since we are using k-fold cross-validation (k=4), we compute the mean of cumulative reward in each test fold. The final average cumulative reward and standard deviation (represented by the error bar) are computed by taking the average across the test folds (4 test folds in total). Bold numbers indicate the winners in each setting. RL agents performed better than the baselines in all settings, but more remarkably in *Any-time learning.*

		Any-time learning (acc_reward = $ALC(\mathcal{T})$)		**Fixed-time learning (acc_reward = $V^*(\mathcal{T})$)**	
		AutoDL	**Artificial**	**AutoDL**	**Artificial**
RL agents	**ddqn**	0.68 ± 0.03	$\mathbf{0.59 \pm 0.01}$	0.84 ± 0.06	$\mathbf{0.83 \pm 0.03}$
	sac	0.62 ± 0.08	0.56 ± 0.03	0.84 ± 0.06	0.77 ± 0.05
	ppo	$\mathbf{0.69 \pm 0.04}$	0.55 ± 0.04	$\mathbf{0.85 \pm 0.05}$	0.77 ± 0.06
Baselines	**freeze-thaw**	0.42 ± 0.05	0.41 ± 0.02	0.73 ± 0.08	0.82 ± 0.02
	average_rank	0.45 ± 0.25	0.57 ± 0.03	0.80 ± 0.06	0.75 ± 0.03
	best_on_samples	0.38 ± 0.08	0.41 ± 0.02	0.55 ± 0.10	0.78 ± 0.02
	random	0.52 ± 0.05	0.44 ± 0.02	0.78 ± 0.04	0.81 ± 0.03

Acknowledgements

We would like to thank Adrien Pavao and Michael Vaccaro for supplying us with the AutoDL meta-dataset.

References

1. Abdulrahman, S., Brazdil, P., van Rijn, J., Vanschoren, J.: Speeding up algorithm selection using average ranking and active testing by introducing runtime. Machine Learning **107** (01 2018)
2. Alexander Smith: Mouse in a maze, `https://videogamehistorian.wordpress.com/tag/mouse-in-a-maze/`, [Online; accessed 06-July-2021]
3. Bensusan, H., Kalousis, A.: Estimating the predictive accuracy of a classifier. In: De Raedt, L., Flach, P. (eds.) Machine Learning: ECML 2001. pp. 25–36. Springer Berlin Heidelberg, Berlin, Heidelberg (2001)
4. Bertinetto, L., Henriques, J.F., Torr, P., Vedaldi, A.: Meta-learning with differentiable closed-form solvers. In: International Conference on Learning Representations (2019), `https://openreview.net/forum?id=HyxnZh0ct7`
5. Brazdil, P.B., Soares, C.: A comparison of ranking methods for classification algorithm selection. In: López de Mántaras, R., Plaza, E. (eds.) Machine Learning: ECML 2000. pp. 63–75. Springer Berlin Heidelberg, Berlin, Heidelberg (2000)
6. Finn, C., Abbeel, P., Levine, S.: Model-agnostic meta-learning for fast adaptation of deep networks. In: Proceedings of the 34th International Conference on Machine Learning - Volume 70. p. 1126–1135. ICML'17, JMLR.org (2017)
7. Fusi, N., Sheth, R., Elibol, M.: Probabilistic matrix factorization for automated machine learning. In: Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., Garnett, R. (eds.) Advances in Neural Information Processing Systems. vol. 31. Curran Associates, Inc. (2018), `https://proceedings.neurips.cc/paper/2018/file/b59a51a3c0bf9c5228fde841714f523a-Paper.pdf`
8. Guerra, S.B., Prudêncio, R.B.C., Ludermir, T.B.: Predicting the performance of learning algorithms using support vector machines as meta-regressors. In: Kůrková, V., Neruda, R., Koutník, J. (eds.) Artificial Neural Networks - ICANN 2008. pp. 523–532. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
9. Haarnoja, T., Zhou, A., Abbeel, P., Levine, S.: Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In: Dy, J., Krause, A. (eds.) Proceedings of the 35th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 80, pp. 1861–1870. PMLR (10–15 Jul 2018), `http://proceedings.mlr.press/v80/haarnoja18b.html`
10. Hasselt, H.v., Guez, A., Silver, D.: Deep reinforcement learning with double q-learning. In: Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence. p. 2094–2100. AAAI'16, AAAI Press (2016)
11. Kopf, C., Taylor, C.: Meta-analysis: From data characterisation for meta-learning to meta-regression (2000)
12. Lee, K., Maji, S., Ravichandran, A., Soatto, S.: Meta-learning with differentiable convex optimization. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 10649–10657 (2019)
13. Leite, R., Brazdil, P.: Predicting relative performance of classifiers from samples. In: Proceedings of the 22nd International Conference on Machine Learning. p. 497–503. ICML '05, Association for Computing Machinery, New York, NY, USA (2005), `https://optdoi.org/10.1145/1102351.1102414`
14. Leite, R., Brazdil, P.: An iterative process for building learning curves and predicting relative performance of classifiers. In: Proceedings of the Aritficial Intelligence 13th Portuguese Conference on Progress in Artificial Intelligence. p. 87–98. EPIA'07, Springer-Verlag, Berlin, Heidelberg (2007)

15. Leite, R., Brazdil, P.: Active testing strategy to predict the best classification algorithm via sampling and metalearning. p. 309–314. IOS Press, NLD (2010)
16. Leite, R., Brazdil, P., Vanschoren, J.: Selecting classification algorithms with active testing. vol. 7376, pp. 117–131 (07 2012)
17. Lin, S.: Rank aggregation methods. Wiley Interdisciplinary Reviews: Computational Statistics **2**, 555 – 570 (09 2010)
18. Liu, Z., Guyon, I.: Asymptotic Analysis of Meta-learning as a Recommendation Problem. In: Meta-learning Workshop @ AAAI 2021. Virtual, Canada (Feb 2021)
19. Liu, Z., Pavao, A., Xu, Z., Escalera, S., Ferreira, F., Guyon, I., Hong, S., Hutter, F., Ji, R., Junior, J.C., Li, G., Lindauer, M., Zhipeng, L., Madadi, M., Nierhoff, T., Niu, K., Pan, C., Stoll, D., Treger, S., Jin, W., Wang, P., Wu, C., Xiong, Y., Zela, A., Zhang, Y.: Winning solutions and post-challenge analyses of the chalearn autodl challenge 2019. IEEE Transactions on Pattern Analysis and Machine Intelligence pp. 1–1 (2021)
20. Liu, Z., Pavao, A., Xu, Z., Escalera, S., Ferreira, F., Guyon, I., Hong, S., Hutter, F., Ji, R., Nierhoff, T., Niu, K., Pan, C., Stoll, D., Treguer, S., Wang, J., Wang, P., Wu, C., Xiong, Y.: Winning solutions and post-challenge analyses of the ChaLearn AutoDL challenge 2019. IEEE Transactions on Pattern Analysis and Machine Intelligence p. 17 (2020)
21. Liu, Z., Xu, Z., Rajaa, S., Madadi, M., Junior, J.C.S.J., Escalera, S., Pavao, A., Treguer, S., Tu, W.W., Guyon, I.: Towards automated deep learning: Analysis of the autodl challenge series 2019. In: Escalante, H.J., Hadsell, R. (eds.) Proceedings of the NeurIPS 2019 Competition and Demonstration Track. Proceedings of Machine Learning Research, vol. 123, pp. 242–252. PMLR (08–14 Dec 2020), `http://proceedings.mlr.press/v123/liu20a.html`
22. Misir, M., Sebag, M.: Alors: An algorithm recommender system. Artif. Intell. **244**, 291–314 (2017)
23. Misir, M., Sebag, M.: Algorithm selection as a collaborative filtering problem (12 2013)
24. Nichol, A., Schulman, J.: Reptile: a scalable metalearning algorithm (03 2018)
25. Petrak, J.: Fast subsampling performance estimates for classification algorithm selection. In: Proceedings of the ECML-00 Workshop on Meta-Learning: Building Automatic Advice Strategies for Model Selection and Method Combination. pp. 3–14 (2000)
26. Schulman, J., Wolski, F., Dhariwal, P., Radford, A., Klimov, O.: Proximal policy optimization algorithms (07 2017)
27. Stern, D., Samulowitz, H., Herbrich, R., Graepel, T., Pulina, L., Tacchella, A.: Collaborative expert portfolio management. vol. 1 (12 2010)
28. Sun-Hosoya, L.: Meta-Learning as a Markov Decision Process. Theses, Université Paris Saclay (COmUE) (Dec 2019), `https://hal.archives-ouvertes.fr/tel-02422144`
29. Sun-Hosoya, L., Guyon, I., Sebag, M.: Activmetal: Algorithm recommendation with active meta learning. In: IAL@PKDD/ECML (2018)
30. Swersky, K., Snoek, J., Adams, R.: Freeze-thaw bayesian optimization (06 2014)
31. Vanschoren, J.: Meta-learning: A survey. ArXiv **abs/1810.03548** (2018)
32. Wikipedia contributors: Battleship (game), `https://en.wikipedia.org/wiki/Battleship_(game)`, [Online; accessed 06-July-2021]
33. Wikipedia contributors: Minesweeper (video game), `https://en.wikipedia.org/wiki/Minesweeper_(video_game)`, [Online; accessed 06-July-2021]
34. Wikipedia contributors: Pac-man, `https://en.wikipedia.org/wiki/Pac-Man`, [Online; accessed 02-July-2021]

35. Yang, C., Akimoto, Y., Kim, D.W., Udell, M.: Oboe: Collaborative filtering for automl model selection. In: Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining. p. 1173–1183. KDD '19, Association for Computing Machinery, New York, NY, USA (2019), https://optdoi.org/10.1145/3292500.3330909

Uncertainty and Utility Sampling with Pre-Clustering

Zhixin Huang, Yujiang He, Stephan Vogt, and Bernhard Sick

Intelligent Embedded Systems, University of Kassel, Kassel, Germany
{zhixin.huang,yujiang.he,stephan.vogt,bsick}@uni-kassel.de

Abstract. Uncertainty sampling is one of the main approaches in deep active learning. In the early phase of uncertainty sampling, uninformative instances are usually selected due to missing exploration of the data space. This can result in a poor quality model leading to poorer acquisitions and further leading to a poorer model. Clustering algorithms can analyze large amounts of unlabeled data in an unsupervised way. A cluster center can be seen as the representative of its cluster and is often highly useful for querying the label from the oracle. Therefore, we propose an algorithm that enables the model to explore the data space at the initial stage using pre-clustering, and enhances the exploration of uncertainty sampling continually based on a combination of uncertainty and utility metrics. The preliminary experimental results show that the proposed algorithm supports balance and imbalanced data scenarios. Besides, our algorithm can achieve a higher classification accuracy compared to baselines methods, even under fewer annotations.

Keywords: Active Learning · Deep Active Learning · Bayesian Neural Network · Uncertainty Sampling · Clustering.

1 Introduction

Deep learning (DL) has a strong learning ability to process high-dimensional data and extract features automatically [24], while DL is often very greedy for data [11]. Active learning is concerned with reducing annotation costs effectively and ensuring a predetermined level of accuracy. However, a major challenge in AL is its lack of scalability to high-dimensional data [29]. Therefore, an approach that combines DL and AL will significantly expand their application potential. This combined approach, referred to as deep active learning (DAL), mainly contains two parts: the AL query strategy on the unlabeled data set and the DL model training [24]. In the pool-based AL scenario, the selection strategy chooses the best sample based on the evaluation and ranking of the entire large data set. The annotated samples are used to train the model and improve the data acquisition for the next AL iteration. The uncertainty-based approach is one of the most common pool-based methods in the application, because it is simple in form and has low computational complexity [24]. Many DAL [1, 10, 22, 23] methods use the uncertainty sampling (US) strategy directly. However, there are still two challenges that have to be overcome:

- **Unreliable uncertainty at the initial AL phase** Uninformative instances are usually selected based on unreliable uncertainty due to an unclear sense of the data space at the early stage. This can result in a poor quality model leading to poorer acquisition and further leading to a poorer model [3].
- **Uncertainty sampling lacks exploration** For uncertainty sampling in DAL context, [6, 14, 12] utilize batch acquisition and query the top n instances with the highest scores. However, it is likely to select a set of information-rich but similar samples [33]. It leads to insufficient exploration, i.e., the knowledge regarding the data distribution is not fully utilized [24], which makes low DL model training efficiency and high annotation cost.

To address the first challenge, it is crucial to find the most representative instances from the large unlabeled data set at the initial AL phase. The general method [7] is to use random selection (RS) at the beginning of the training process for exploration. However, this method could fail for imbalanced data set because the selected instances are less representative, and most of them locate dense areas [30]. The model can deeply learn the true data space only when sufficient labels of data are available. However, it will increase annotation cost. Unsupervised learning algorithms can analyze large amounts of unlabeled data. For example, the K-Means [26] algorithm is one of the most common clustering algorithms for knowledge discovery in data mining. The cluster information is helpful for AL in two aspects: (1) The instances located in the center of clusters are more representative than the others and should be labeled firstly; (2) Samples in the same cluster are likely to have the same label [21].

For the second challenge, a feasible solution is to use a hybrid query strategy to enhance the exploration of US. The similarity between samples is a method [21, 15] to measure the similarity amongst instances by calculating the feature vectors' distance between each other. Similar to US, these algorithms are often only good at exploitation, i.e., the learners tend to only focus on instances near the current decision boundary [24]. But in the opposite direction, we can also utilize the similarity to exclude similar samples. After sorting a batch of instances based on the uncertainty through US, we could filter out similar instances to improve the exploration of selection strategy.

To overcome the challenges mentioned above, the two core ideas of our proposed algorithm are: (1) At the initial phase, we label the instances closest to cluster centers to train the model for estimating reliable uncertainty. (2) The selection of the most informative instance depends on two strategies, uncertainty and utility. The uncertainty evaluates the epistemic uncertainty of Bayesian Neural Network (BNN) [6, 7] to an instance. The utility filters out the instances which are similar to the already labeled instances. Since US lacks exploration in the data space, the utility metric helps the model discover some valuable instances far away from the current decision boundary. Therefore, we propose our algorithm **U**ncertainty and **U**tility sampling with **P**re-**C**lustering (UUPC). Compared to the baselines, our algorithm can achieve a higher classification accuracy under fewer annotations.

The remainder of this article starts with a summary of the related work. The details of the algorithm and the experiments are introduced in Section 3 and 4 respectively. This article is closed with a conclusion and an outlook on our future work in this field.

2 Related Work

The uncertainty-based query strategies (e.g., Margin Sampling and Entropy) in the DAL scenario are widely used [6, 14, 12] because it is convenient to combine with the output of the DL model. Traditional DL requires a large amount of labeled data to obtain reliable uncertainty estimation. In the DAL scenario with large unlabeled data, epistemic uncertainty is particularly valuable because it allows the model to assess its lack of knowledge. For this reason, a method that combines deep Bayesian neural network with US has been proposed [7, 12, 22]. However, as analyzed in Section 1, US could select uninformative instances at the initial AL phase and lack exploration. Therefore, some hybrid query strategies are developed [32, 34], taking into account the uncertainty and diversity of samples. Exploration-P [32] utilizes a deep neural network to obtain the uncertainty and the similarity between the samples. Besides, this method uses RS strategy for exploration purposes in the early AL phase. The combination of AL and K-means clustering has been researched in previous works [13, 21] to find the most representative instances. DBAL [34] presents a hybrid query approach that utilizes the K-means clustering algorithm to explore the diversity of instances in each mini-batch. Contrary to [34], which performs clustering in each AL iteration, our approach annotates labels based on cluster centers only at the initial AL phase to pretrain the BNN model. Thus, it can avoid labeling samples repeatedly in the same cluster. Similar to select the most representative instances by clustering, the core set approach is also a representative query strategy. The basic idea is constructing a core set to represent the distribution of the feature space of the entire original data set, thereby reducing the labeling cost of AL [27, 31]. However, the core-set approach requires building a large distance matrix on the unlabeled data set, the search process is computationally expensive especial on the large data set [2].

3 Problem Formulation and Algorithms

In the general classification, one sample is described by $\mathbf{x} \in \mathcal{X}$ and its label from C classes with a corresponding label $y \in \mathcal{Y} = \{1, \cdots, C\}$. The clustering information can be described explicitly by introducing the cluster label $k \in \{1, \cdots, K\}$, where K is the number of clusters in the data. In the pool-based AL, we define $\mathcal{U} = \{\mathbf{x}_1, \cdots, \mathbf{x}_N\}$ as an unlabeled set with N samples. Labels are not available at the beginning but can be annotated by the oracle. The query strategy selects an instance $\mathbf{x} \in \mathcal{U}$ and asks the oracle for the corresponding label $y \in \mathcal{Y}$. The newly labeled instance is removed from the unlabeled set $\mathcal{U} \leftarrow \mathcal{U} \backslash \mathbf{x}$. We add the instance with its label to the labeled set $\mathcal{L} \leftarrow \mathcal{L} \cup (\mathbf{x}, y)$, and train supervised learning models such as SVM and DNN on $\mathcal{L}$.

3.1 Pre-Clustering at initial AL Phase

Selecting the most representative instances from the unlabeled data by labeling cluster centers is heavily dependent on the quality of clustering results. In K-Means, the crucial parameter that affects the goodness of clustering results is the number of clusters, which should be optimized. The evaluation without any labels must be performed using the model itself. The elbow method [16] is the most popular heuristic approach, which calculates the sum of squared distances (SD) from each point to its assigned center. The unsupervised evaluation scores such as Silhouette Coefficient (SC) [25], Calinski-Harabasz Index (CHI) [4] and Davies-Bouldin Index (DBI) [9] could also be applied to the elbow method. We will calculate multiple cluster scores to determine the optimal number of clusters K_o. To optimize SC and CHI, we have to maximize the scores, while lower SD and DBI indicate a model with better defined clusters so they must be minimized. We take the reciprocal of SC and CHI to unify the optimization direction. The weighted score of pre-clustering (PC) is calculated by following:

$$\begin{aligned} \underset{\mathrm{PC}}{\mathrm{Score}}(K,\mathcal{U}) = \; & \alpha_1 \,\mathrm{SD}(K,\mathcal{U}) + \alpha_2 \,\mathrm{DBI}(K,\mathcal{U}) \\ & + \alpha_3 \frac{1}{\mathrm{SC}(K,\mathcal{U})} + \alpha_4 \frac{1}{\mathrm{CHI}(K,\mathcal{U})} + \lambda K. \end{aligned} \tag{1}$$

The weights of each score are $\alpha_{1,\cdots,4}$, and the sum is 1. The $\alpha_{1,\cdots,4}$ could be selected by expert knowledge, or in the absence of detailed expert knowledge, like in the experiment in Section 4, all weights are selected to be the same value. In our definition, K must be equal or greater than C. For example, MNIST [19] requires at least 10 clusters, one per class. K_{max} indicates the maximum budget of annotations at the initial AL phase, and we expect that $C < K_{max} \ll N$. Since the above four cluster evaluation scores have different scales, in practice, we calculate a set for each type of score (SD, DBI and reciprocal of SC and CHI) from C to K_{max} and normalize each set to 0-1 range. Then we add the four scores to obtain a set of $\underset{\mathrm{PC}}{\mathrm{Score}}(K,\mathcal{U})$, where $K \in \{C, \ldots, K_{max}\}$. The larger the K, the smaller the $\underset{\mathrm{PC}}{\mathrm{Score}}$, which means that the more refined clustering. However, the labeling cost must be considered because the oracle has to annotate every instance closest to the center in each cluster. Therefore we append λK into $\underset{\mathrm{PC}}{\mathrm{Score}}(K,\mathcal{U})$ as the regularization, where λ is the weight of regularization and proportional to the cost of an annotation. Setting a proper value of λ is dependent on the application scenario and requires expert experience. The Bayesian information criterion (BIC) and the Akaike information criterion (AIC) could determine the appropriate number of clusters without tuning regularization [28, 8]. But they can be applied only if we extend the clustering algorithm beyond K-Means to Gaussian Mixture Model (GMM). Since this paper utilizes pre-clustering by K-Means, BIC and AIC will be researched in future work. The optimal number of clusters K_o can be described as follows:

$$K_o = \underset{K \in \{C, \cdots, K_{max}\}}{\operatorname{argmin}} \underset{\mathrm{PC}}{\operatorname{Score}} (K, \mathcal{U}) \tag{2}$$

Assume that the information about the class label y is encoded in the cluster k. The set of elements in cluster k is $\mathbf{c}_k$. Once the data probability distribution of clusters $p(\mathbf{x} \in \mathbf{c}_k)$ and the class y_k of each cluster center $\mathbf{x}_k$ are known, we can infer the probability distribution of class $p(y|\mathbf{x} \in \mathbf{c}_k)$ with respect to all samples in $\mathbf{c}_k$ [21]. However, using the cluster center to annotate all instances' labels is not reliable because the samples located at the intersection of clusters are easily misclassified. In contrast with refining smaller clusters [21], our method only uses the pre-clustering to pretrain the model. In detail, we only label the instances closest to each cluster center by oracle $p(y_k|\mathbf{x}_k, k)$ and put them into the labeled data set $\mathcal{L} = \{(\mathbf{x}_j, y_j) \mid j \in \{1, \cdots, K_o\}\}$, where K_o is optimal number of clusters. At the initial phase of our approach, the BNN will learn the initial labeled data set to get optimal posterior parameters for reliable uncertainty estimation. Then the oracle will label the most informative instances based on the combination of the following two selection functions: uncertainty and utility.

3.2 Uncertainty-Utility Selection Strategy at AL Phase

The BNN can be defined as $f(\mathbf{x}, \boldsymbol{\theta})$. $p(\boldsymbol{\theta})$ where $\boldsymbol{\theta} \in \Theta$ is a prior on the parameter space Θ. The likelihood $p(y|\mathbf{x}, \boldsymbol{\theta})$ is determined by softmax $(f(\mathbf{x}, \boldsymbol{\theta}))$. The goal is to obtain the posterior distribution over $\boldsymbol{\theta}$ from labeled training set $\mathcal{L}$:

$$p(\boldsymbol{\theta}|\mathcal{L}) = \frac{p(\mathcal{L}|\boldsymbol{\theta})\, p(\boldsymbol{\theta})}{p(\mathcal{L})} \tag{3}$$

The $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T$ are sampled T times to get an monte carlo estimate of the predictive probabilitiy distribution on the label y as the average regarding a new unlabeled instance $\mathbf{x}^* \in U$:

$$\hat{p}(y|\mathbf{x}^*, \mathcal{L}) = \frac{1}{T} \sum_{t=1}^{T} p(y|\mathbf{x}^*, \mathcal{L}, \boldsymbol{\theta}_t) \tag{4}$$

Equation 4 describes the general uncertainty estimation of BNN, and it includes both the epistemic and aleatoric uncertainty of the prediction y. In our case, we calculate the entropy over the predicted class probabilities of a new instance to estimate the uncertainty score as given in the numerator of Eq. 5. In each AL iteration, the scores of instances in $\mathcal{U}$ are normalized into a 0–1 range, where 1 is the most uncertain score, indicating that being annotated is often very useful. The function Uncr $(\mathbf{x}^*)$ can evaluate the uncertainty score for each instance in $\mathcal{U}$:

$$\operatorname{Uncr}(\mathbf{x}^*) = \frac{-\sum_{c}^{C} \hat{p}(y = c|\mathbf{x}^*, \mathcal{L}) \log_2 (\hat{p}(y = c|\mathbf{x}^*, \mathcal{L}))}{\log_2 (C)}. \tag{5}$$

As mentioned in Section 2, the uncertainty metric requires to be enhanced with exploration of the data space. Although in the initial stage, we use pre-clustering to help BNN to obtain reliable uncertainty estimations quickly, some valuable instances are far from the current existing decision boundary. Therefore, we define a utility metric to enhance the exploration of US continually. We define the Euclidean distances between two instances $\mathbf{x}_1$ and $\mathbf{x}_2$ as $\mathrm{Dis}\left(\mathbf{x}_1, \mathbf{x}_2\right)$. The similarity between the instance $\mathbf{x}^*$ to class c is defined as the median distances of $\mathbf{x}^*$ to all instances of c in the $\mathcal{L}$. The formulation can be written as:

$$\mathrm{Sim}\left(\mathbf{x}^*, c\right) = \mathrm{median}(\{\mathrm{Dis}\left(\mathbf{x}^*, \mathbf{x}\right), \text{ where } (\mathbf{x}, y) \in \mathcal{L} \text{ and } y = c\}). \tag{6}$$

The standard deviation of the similarities between the instance and each class represents the trend of which class it belongs to. The higher standard deviation indicates the instance is likely to be classified to one single class. When the standard deviation is lower, the instance is located in the intersection of multiple classes, and annotation by the oracle could be more beneficial. For a paired comparison with uncertainty, we transfer the optimization task of this score into a maximization problem. The scale of uncertainty score is 0-1. Hence in practice, we calculate the utility score of each instance in a batch and normalize the entire batch of utility scores to the same scale. Eq. 7 shows the method of calculating the utility of a single instance $\mathbf{x}^*$.

$$\mathrm{Utility}(\mathbf{x}^*) = \frac{1}{\mathrm{std}\left(\{\mathrm{Sim}\left(\mathbf{x}^*, c_1\right), \cdots \mathrm{Sim}\left(\mathbf{x}^*, c_C\right)\}\right)} \tag{7}$$

Uncertainty-utility (UU) score is defined as follows:

$$\underset{\mathrm{UU}}{\mathrm{Score}}\left(x*\right) = \gamma_1 \,\mathrm{Uncr}\left(\mathbf{x}^*\right) + \gamma_2 \,\mathrm{Utility}\left(\mathbf{x}^*\right) \tag{8}$$

where γ_1 and γ_2 are in 0-1 range and control the weights of two selection metrics separately. The weights could be selected by expert knowledge, or in the absence of detailed expert knowledge, γ_1 and γ_2 are each selected equal to 1. The higher score, indicating the more worthy of being annotated.

3.3 Batch-based UUPC Algorithm

With batch training, our method could have more efficient training on large data sets: (1) Clustering, such as K-Means, passes through the entire data set to obtain the centers. The training process is time-consuming, which is proportional to the amount of data. The mini-batch-based K-Means [26] uses a batch-based method to cluster large data sets to reduce computation costs. (2) For traditional uncertainty sampling, each iteration requires uncertainty estimation for all instances in $\mathcal{U}$. In DAL scenario, we use batch-based sample querying to improve training efficiency [24].

At each acquisition step, we score a batch of candidate unlabeled samples $\mathcal{B} \subseteq \mathcal{U}$, where $\mathcal{B} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_b\}$ and b refers to the batch size. Based on the $\underset{\mathrm{UU}}{\mathrm{Score}}$,

Algorithm 1 UUPC Algorithm for Batch Training

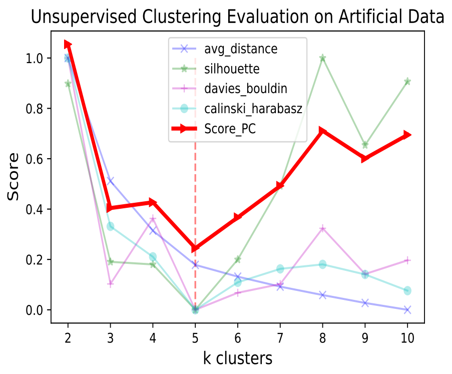
Input: Unlabeled data set $\mathcal{U} \leftarrow \mathcal{X}$, initial labeled set $\mathcal{L} \leftarrow \emptyset$, one batch data $\mathcal{B} \subseteq \mathcal{U}$ with b samples is selected randomly, the process of batch sampling is described as BatchSampling$(\mathcal{U}, b)$, the maximum number of AL iterations for pre-clustering phase B_{pc} and for UU sampling phase B_{uu}, N_{uu} instances are annotated per batch.
Output: Optimized number of cluster K_o, labeled data set $\mathcal{L}$, BNN model $f(\mathbf{x}, \boldsymbol{\theta})$

```
 1: $K_o \leftarrow \underset{K \in \{C, \cdots K_{max}\}}{\operatorname{argmin}} \underset{\text{PC}}{\text{Score}}(K, \mathcal{U})$
 2: $iter \leftarrow 0$
 3: while $iter < B_{pc}$ do
 4:     $\mathcal{B}^{iter} \leftarrow \text{BatchSampling}(\mathcal{U}, b)$
 5:     $\{\mathbf{x}_1^{iter}, \cdots, \mathbf{x}_k^{iter}\} \leftarrow \text{K-Means}(\mathcal{B}^{iter}, K_o)$
 6:     if $\{\mathbf{x}_1^{iter}, \cdots, \mathbf{x}_k^{iter}\} == \{\mathbf{x}_1^{iter-1}, \cdots, \mathbf{x}_k^{iter-1}\}$ then
 7:         Break
 8:     $iter \leftarrow iter + 1$
 9: $\mathcal{L} \leftarrow \{(\mathbf{x}_1, y_1), \cdots, (\mathbf{x}_k, y_k)\} \leftarrow \text{Labeling}(\{\mathbf{x}_1, \cdots, \mathbf{x}_k\})$
10: $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta_T}\} \leftarrow \text{Training}(f(\mathbf{x}, \boldsymbol{\theta}), y)$, where $(\mathbf{x}, y) \in \mathcal{L}$
11: $iter \leftarrow 0$
12: while $iter < B_{uu}$ do
13:     $\mathcal{B}^{iter} \leftarrow \text{BatchSampling}(\mathcal{U}, b)$
14:     $\mathcal{S} \leftarrow \emptyset$
15:     while $i < N_{uu}$ do
16:         $\mathbf{x}_i^* \leftarrow \operatorname{argmax} \underset{\text{UU}}{\text{Score}}(\mathbf{x})$, where $\mathbf{x} \in \mathcal{B}^{iter} \backslash \mathcal{S}$
17:         $\mathcal{S} \leftarrow \mathcal{S} \cup \mathbf{x}_i^*$
18:     $\mathcal{L} \leftarrow \mathcal{L} \cup \text{Labeling}(\mathcal{S})$
19:     $\mathcal{U} \leftarrow \mathcal{U} \backslash \mathcal{S}$
20:     $\{\boldsymbol{\theta}_1, \cdots, \boldsymbol{\theta_T}\} \leftarrow \text{Training}(f(\mathbf{x}, \boldsymbol{\theta}))$, where $\mathbf{x} \in \mathcal{L}$
21:     if $\mathcal{U} == \emptyset$ then
22:         Break
23:     $iter \leftarrow iter + 1$
```

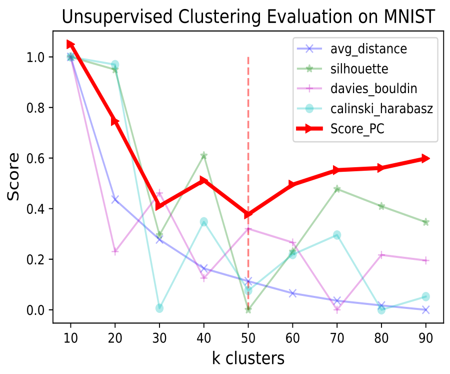
we select the top n candidate instances with the highest scores $\mathcal{S} = \{\mathbf{x}_1^*, \cdots, \mathbf{x}_n^*\}$ where $n \leq b$. This problem can be formulated as follows:

$$\mathbf{x}_i^* = \underset{\mathbf{x} \in \mathcal{B} \backslash \{\mathbf{x}_j^* | j < i\}}{\operatorname{argmax}} \underset{\text{UU}}{\text{Score}}(\mathbf{x}) \tag{9}$$

The UUPC algorithm is shown in Alg. 1. In line 1 of Alg. 1, we select the optimized number of clusters K_o using Eq. 2. In lines 2-8, we choose batches randomly to train the mini-batch-based K-Means model until the positions of cluster centers are not changed. In line 9, the instances, which are the closest to the cluster centers, will be annotated by the oracle and moved into $\mathcal{L}$. Line 10 means training the BNN based on $\mathcal{L}$ to help the model understand the data space at initial AL phase. In lines 11-23, we calculate $\underset{\text{UU}}{\text{Score}}$ (see Eq. 8) on the batches data iteratively and annotate the top N_{uu} instances per batch. The annotated instances are moved to $\mathcal{L}$ to update the BNN model. We stop the process when the budget is exhausted.



(a) λ=0.05, optimal $K_o = 5$



(b) λ=0.005, optimal $K_o = 50$

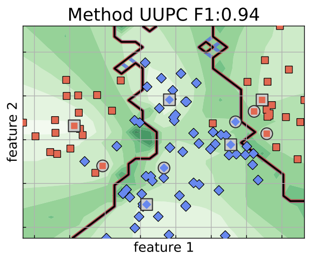
Fig. 1: Unsupervised cluster number evaluation by $\underset{\text{PC}}{\text{Score}}$ on artificial data set and MNIST using Eqs. 1 and 2. The four weights $\alpha_{1,\cdots,4}$ of $\underset{\text{PC}}{\text{Score}}$ are set to the same value of 0.25. The optimal number of clusters K_o locates at the lowest value of $\underset{\text{PC}}{\text{Score}}$. The red vertical dash line indicates the position of the optimal K_o.

4 Experimental Evaluation

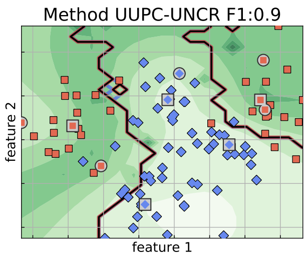
To evaluate the quantitative performance of UUPC, we conduct experiments on artificial and real-world data sets. The following selection algorithms are compared. Besides random selection (RS) from a batch of instances and uncertainty sampling with entropy (US), we also use **R**andom **S**ampling strategies at the initial AL phase before **U**ncertainty **S**ampling (RSUS). For UUPC, K_o instances are selected by pre-clustering at initial AL phase. In order to make a fair comparison between our approach and RSUS, K_o instances are randomly selected as initial $\mathcal{L}$ in the RSUS method. To verify the utility metric, we conduct another strategy UUPC-UNCR, where only the uncertainty is considered, to assess the importance of the utility metric. For UUPC, we set the weights empirically in the $\underset{\text{UU}}{\text{Score}}$ as $\gamma_1 = 1.0$ and $\gamma_2 = 0.7$. In these experiments, we use a simple Bayesian dropout approximation neural network with multiple fully connected layers: 2 dense hidden layers with 250 and 100 units, ReLU activation and dropout, and an output layer. The dropout probabilities are set to 0.3 and 0.5 respectively. The $\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_T$ are sampled ten times to obtain the average probability distribution on the label for each candidate instance, i.e., T is set to 10 in Eq. 4.

4.1 Artificial Data Set

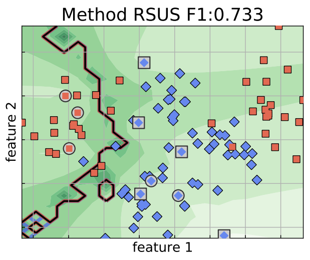
The first experiment is inspired by [17]. Based on a low dimensional and small artificial data set that could visually show the acquisition behavior of different



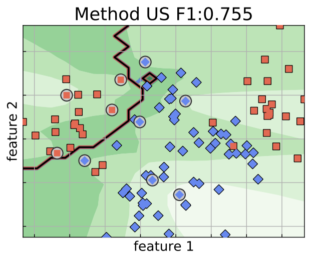
(a) UUPC



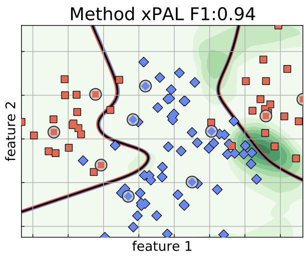
(b) UUPC-UNCR



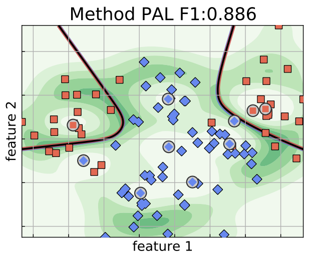
(c) The worst case of RSUS



(d) US



(e) xPAL



(f) PAL

Fig. 2: Visualization of acquisition behavior for different selection strategies on artificial data set. The green color indicates how useful a selection strategy considers a region. Darker areas are considered more valuable than brighter areas. The corresponding selection strategy has selected ten labeled instances marked with gray circles or squares. For UUPC, UUPC-UNCR and RSUS, the first selected five instances at AL initial phase are marked as gray squares. Thereby, one can see the current decision boundaries illustrated by black lines and how the usefulness is spatially distributed to select the next instance for labeling. The artificial data generation and visualization method are inspired by [17].

selection strategies. Through visualization, the performance of UUPC could be visually verified when it utilizes pre-clustering in the initial stage of AL and later selects samples through $\underset{\text{UU}}{\text{Score}}$. We also use F1 scores to quantitatively check whether our proposed method can outperform other baseline methods.

The artificial imbalanced data set contains 100 two-dimensional instances with two classes (60 blue diamonds and 40 red rectangles). We put the whole artificial data set as one batch and select an instance with most information from $\mathcal{U}$ at each AL iteration. One side the data size is too tiny another side it can compare with other traditional AL algorithms. Fig. 2d shows that US only has one unilateral decision boundary on the left side, which lacks exploration. The result of RS is not shown in Fig. 2 because it is unstable and entirely depends on random seeds. The optimized number of pre-clustering K_o is 5 (see Fig. 1a). For UUPC, UUPC-UNCR and RSUS, the initial selected five instances marked as gray squares are distributed in Figs 2a, 2b, and 2c respectively. In the UUPC and UUPC-UNCR methods, the initially selected instances are located in the five cluster centers representing the whole data space. Selecting the most representative instances could help the BNN model to estimate reliable uncertainty. However, similar to RS, the initial random selection strategy in RSUS relies on random seeds. Fig. 2c illustrates one of the worst cases of RSUS because all initial randomly selected instances belong to the blue diamonds class. This results in a poor quality model leading to poorer acquisition. The results in Figs. 2a and 2b prove that UUPC could increase the F1 score by 4% compares with UUPC-UNCR. Furthermore, we compare the results of the other two methods xPAL [17] and PAL [18] visually[1]. Since these two methods did not use BNN as a classifier, it might affect the output of their selection strategy. Here we only simply show the distribution of the labeled points. As shown in Fig 2e, our method could get similar F1 score as xPAL.

4.2 Real Data Set: MNIST

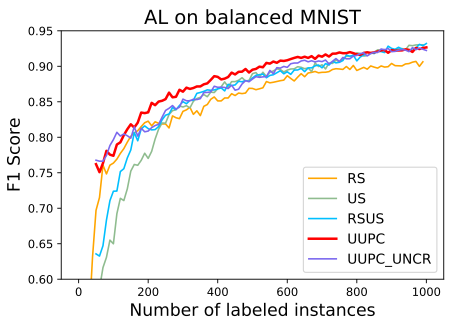
In the above experiment, we visualize the behavior of different selection strategies on low-dimensional artificial data. This experiment aims at evaluating the UUPC performance on real-world balanced and imbalanced data sets with high dimensions. MNIST [2] [19] data set includes 10 handwritten digits. The data set contains 20,000 training images and 10,000 testing images with the shape 28×28. As shown in Fig 1b, the hyperparameter of pre-clustering K_o is 50. The batch size is 1000, and we select the top 10 highest-scoring instances from each batch. We repeat the following experiments 20 times and evaluate the performance of different methods on the test data sets through the F1 score.

Fig. 3a illustrates the F1 scores of the test set with the different amount of annotations on balanced data set. We set the whole training set of MNIST to $\mathcal{U}$ and label 5% (1000 annotations) unlabeled instances in $\mathcal{U}$. Same as what [3, 17, 24] pointed out, US does even worse than RS when the number of annotations is smaller than 200 due to unreliable uncertainty estimations. UUPC and UUPC-UNCR outperform other methods at the initial phase because of pre-clustering. Due to only uncertainty is considered in UUPC-UNCR, the advantage of pre-clustering decays gradually after 200 annotations. When the number of labeled instances exceeds 250, the F1 score of RS increases slower than UUPC and US. It indicates that the uncertainty estimation given by BNN gets more and more important once sufficient annotations are available. UUPC could keep a higher F1 score, which is up to 4.5% higher than other baseline methods, until the number of the annotated samples is greater than 800.

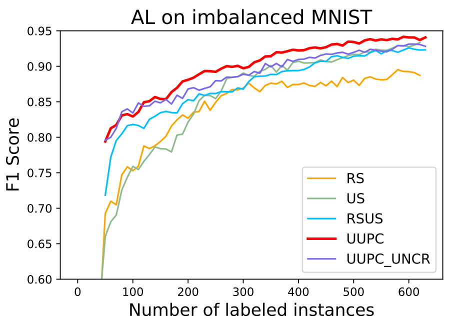
Imbalanced data sets are very common in real-world applications. As a preliminary experiment, we randomly drop 75% of samples of digits 5, 6, 7, 8, 9 in the training and test set, to assess the performance of the methods in

[1] The algorithms of xPAL and PAL, as well as visualization presented in Fig. 2, are implemented by Kottke et al. https://github.com/dakot/probal.

[2] Obtained from https://colab.research.google.com.



(a) $\mathcal{U}$ is balanced MNIST.



(b) $\mathcal{U}$ is imbalanced MNIST.

Fig. 3: Learning curves for MNIST data sets. Each plot shows the multi-class F1 score of UUPC and the competing algorithms on the test images after annotating up to 5% instances from the balanced or imbalanced MNIST unlabeled data set. The learning curve that reaches a high F1 score fast is considered best.

imbalanced data set. Similar to the experiment in the balanced data set, 5% (630 annotations) unlabeled instances in $\mathcal{U}$ will be annotated. Since UUPC and UUPC-UNCR use pre-clustering, the F1 score in the initial phase is still higher than other methods presented in Fig. 3b. Due to selected instances are less representative, and most of them belong to majority classes, the F1 score of RS almost stops increasing after 300 annotations. It is worth noting that when the number of annotations is less than 150, the F1 score of UUPC-UNCR is slightly higher than that of UUPC, which means that when there are fewer annotations, the utility criterion may introduce uninformative samples. One solution is to set utility weight γ_2 to 0 at the initial stage and increase its value corresponding to the number of annotations dynamically. When the size of $\mathcal{L}$ is greater than 150, the utility could enhance the exploration of the selection strategy and increase classification accuracy significantly. Compared with other methods, the F1 score of UUPC is 4.3% higher than other methods on average under the same amount of labeling. In other words, our proposed method reduces the annotation cost by 33.1% on average but achieves the same performance as other baseline methods.

5 Conclusion & Future Work

The direct use of US in DAL could face two main challenges: the unreliable uncertainty estimation in the initial AL phase leads to poor acquisitions and further results in a poorer model, and the lack of exploration of US leads to insufficient diversity of samples. In this article, we propose an effective DAL algorithm UUPC, which enables the model to explore the data space at the initial stage using pre-clustering, and enhance the exploration of uncertainty sampling continually based on a combination of uncertainty and utility metrics. The method is assessed in preliminary experiments. The experimental results show that our method outperforms the baseline methods in balanced and imbalanced data sets under few annotations.

This work can be further researched in these directions: (1) In the current preliminary experiment, we only apply a tiny three-layer linear network and flatten the image data without considering image features. Gal et al. [7] proved that CNN could improve the recognition accuracy under the same number of annotations. It is necessary to extract features through CNN from high-dimensional data in future experiments. (2) The batch-based K-Means algorithm is applied in pre-clustering to improve computational efficiency. It is worth using Autoencoder with CNN layer to reduce dimensionality and extract the most informative features before clustering in further research. (3) At present, we only do preliminary experiments on artificial and MNIST data sets to verify our proposed method's feasibility. Further evaluations are needed on more data sets in the future. Besides, we will compare other existing selection strategies in DAL in further research. (4) UUPC and others mentioned methods above might fail in anomaly detection scenarios. One potential solution is performing isolation forest [20] or DBSCAN [5] at the initial stage of AL to get the rough decision boundary and then refining the result through uncertainty-utility (UU) strategy. (5) The method of obtaining the optimal number of clusters proposed in Subsection 3.1 is still a heuristic algorithm. In different application scenarios, estimating the weights of each sub-score and regularization weight λ in Eq. 1 relies on expert experience. The Bayesian information criterion (BIC) and the Akaike information criterion (AIC) could also determine the appropriate number of clusters [28, 8]. The advantage is that they originally contain regularization and do not require experts to set additional weights.

Acknowledgments

This work is supported within the Digital-Twin-Solar (03EI6024E) project, funded by BMWi: Deutsches Bundesministerium für Wirtschaft und Energie/German Federal Ministry for Economic Affairs and Energy. Special thanks to Daniel Kottke. His paper [17] and corresponding code allow us to understand the details of different AL algorithms. Besides, we compared our proposed method with other AL algorithms based on his visualization code to verify our preliminary idea. Thanks to the colleagues from IES at the University of Kassel, whose reviews and comments have helped improve the manuscript.

References

1. Asghar, N., Poupart, P., Jiang, X., Li, H.: Deep active learning for dialogue generation. arXiv preprint arXiv:1612.03929 (2016)
2. Ash, J.T., Zhang, C., Krishnamurthy, A., Langford, J., Agarwal, A.: Deep batch active learning by diverse, uncertain gradient lower bounds. arXiv preprint arXiv:1906.03671 (2019)

3. Attenberg, J., Provost, F.: Inactive learning? difficulties employing active learning in practice. ACM SIGKDD Explorations Newsletter **12**(2), 36–41 (2011)
4. Caliński, T., Harabasz, J.: A dendrite method for cluster analysis. Communications in Statistics-theory and Methods **3**(1), 1–27 (1974)
5. Ester, M., Kriegel, H.P., Sander, J., Xu, X., et al.: A density-based algorithm for discovering clusters in large spatial databases with noise. In: Kdd. vol. 96, pp. 226–231 (1996)
6. Gal, Y., Ghahramani, Z.: Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In: international conference on machine learning. pp. 1050–1059. PMLR (2016)
7. Gal, Y., Islam, R., Ghahramani, Z.: Deep Bayesian active learning with image data. In: Precup, D., Teh, Y.W. (eds.) Proceedings of the 34th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 70, pp. 1183–1192. PMLR (06–11 Aug 2017), http://proceedings.mlr.press/v70/gal17a.html
8. Grall-Maes, E., Dao, D.T.: Assessing the number of clusters in a mixture model with side-information. In: ICPRAM. pp. 41–47 (2016)
9. Halkidi, M., Batistakis, Y., Vazirgiannis, M.: On clustering validation techniques. Journal of intelligent information systems **17**(2), 107–145 (2001)
10. He, T., Jin, X., Ding, G., Yi, L., Yan, C.: Towards better uncertainty sampling: Active learning with multiple views for deep convolutional neural network. In: 2019 IEEE International Conference on Multimedia and Expo (ICME). pp. 1360–1365. IEEE (2019)
11. Hinton, G.E., Srivastava, N., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.R.: Improving neural networks by preventing co-adaptation of feature detectors. arXiv preprint arXiv:1207.0580 (2012)
12. Houlsby, N., Huszár, F., Ghahramani, Z., Lengyel, M.: Bayesian active learning for classification and preference learning. arXiv preprint arXiv:1112.5745 (2011)
13. Huang, S.J., Jin, R., Zhou, Z.H.: Active learning by querying informative and informative examples. IEEE Transactions on Pattern Analysis and Machine Intelligence **36**(10), 1936–1949 (2014)
14. Janz, D., van der Westhuizen, J., Hernández-Lobato, J.M.: Actively learning what makes a discrete sequence valid. arXiv preprint arXiv:1708.04465 (2017)
15. Joshi, A.J., Porikli, F., Papanikolopoulos, N.: Multi-class batch-mode active learning for image classification. In: 2010 IEEE international conference on robotics and automation. pp. 1873–1878. IEEE (2010)
16. Ketchen, D.J., Shook, C.L.: The application of cluster analysis in strategic management research: an analysis and critique. Strategic management journal **17**(6), 441–458 (1996)
17. Kottke, D., Herde, M., Sandrock, C., Huseljic, D., Krempl, G., Sick, B.: Toward optimal probabilistic active learning using a bayesian approach. Machine Learning pp. 1–33 (2021)
18. Kottke, D., Krempl, G., Lang, D., Teschner, J., Spiliopoulou, M.: Multi-class probabilistic active learning. In: Proceedings of the Twenty-second European Conference on Artificial Intelligence. pp. 586–594 (2016)
19. LeCun, Y., Cortes, C.: MNIST handwritten digit database (2010), http://yann.lecun.com/exdb/mnist/
20. Liu, F.T., Ting, K.M., Zhou, Z.H.: Isolation forest. In: 2008 Eighth IEEE International Conference on Data Mining. pp. 413–422 (2008). https://doi.org/10.1109/ICDM.2008.17

21. Nguyen, H.T., Smeulders, A.: Active learning using pre-clustering. In: Proceedings of the twenty-first international conference on Machine learning. p. 79 (2004)
22. Ostapuk, N., Yang, J., Cudré-Mauroux, P.: Activelink: deep active learning for link prediction in knowledge graphs. In: The World Wide Web Conference. pp. 1398–1408 (2019)
23. Ranganathan, H., Venkateswara, H., Chakraborty, S., Panchanathan, S.: Deep active learning for image classification. In: 2017 IEEE International Conference on Image Processing (ICIP). pp. 3934–3938. IEEE (2017)
24. Ren, P., Xiao, Y., Chang, X., Huang, P.Y., Li, Z., Chen, X., Wang, X.: A survey of deep active learning. arXiv preprint arXiv:2009.00236 (2020)
25. Rousseeuw, P.J.: Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics **20**, 53–65 (1987)
26. Sculley, D.: Web-scale k-means clustering. In: Proceedings of the 19th international conference on World wide web. pp. 1177–1178 (2010)
27. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. arXiv preprint arXiv:1708.00489 (2017)
28. Teklehaymanot, F.K., Muma, M., Zoubir, A.M.: Bayesian cluster enumeration criterion for unsupervised learning. IEEE Transactions on Signal Processing **66**(20), 5392–5406 (2018)
29. Tong, S.: Active learning: theory and applications. Stanford University (2001)
30. Wang, D., Shang, Y.: A new active labeling method for deep learning. In: 2014 International joint conference on neural networks (IJCNN). pp. 112–119. IEEE (2014)
31. Wolf, G.W.: Facility location: concepts, models, algorithms and case studies. series: Contributions to management science: edited by zanjirani farahani, reza and hekmatfar, masoud, heidelberg, germany, physica-verlag, 2009, 549 pp. (2011)
32. Yin, C., Qian, B., Cao, S., Li, X., Wei, J., Zheng, Q., Davidson, I.: Deep similarity-based batch mode active learning with exploration-exploitation. In: 2017 IEEE International Conference on Data Mining (ICDM). pp. 575–584. IEEE (2017)
33. Zhdanov, F.: Diverse mini-batch active learning. arXiv preprint arXiv:1901.05954 (2019)
34. Zhdanov, F.: Diverse mini-batch active learning. arXiv preprint arXiv:1901.05954 (2019)

Evidential Nearest Neighbours in Active Learning

Daniel Zhu[1], Arnaud Martin[1], Yolande Le Gall[1], Jean-Christophe Dubois[1], and Vincent Lemaire[2]

[1] Univ Rennes, CNRS, IRISA, DRUID, rue E. Branly, Lannion
`firstname.lastname@irisa.fr`
[2] Orange Labs, France, `vincent.lemaire@orange.com`

Abstract. Active learning is a subfield of machine learning which allows to reduce the amount of data necessary to train a classifier. The training set is built in an iterative way such that only the most significant and informative data are used and labeled by an external person called oracle. It is furthermore possible to use active learning with the theory of belief functions in order to take erroneous labels due to the oracle's uncertainty and imprecision into account in order to limit their influence on the classifier's performance. In this article, we compare the classifier of the k nearest neighbours (kNN) to a variant based on belief functions from the theory of belief functions (EkNN), in a situation where some labels have been noised in order to model uncertain labels. We show that although the superiority of EkNN over kNN is not systematic, there are some interesting and modest results supporting the relevance of belief functions in active learning.

Keywords: Active Learning · Belief Functions · Theory of Dempster-Shafer · Nearest Neighbours

1 Introduction

In supervised machine learning, the size of the training set, *i.e.* the number of labeled examples, is often correlated to the performance of the learned model. Although having access to large database is no longer difficult nowadays, labeling the data remains an expensive task, especially when the application domain requires some expertise. Active learning offers a solution to this issue by reducing the number of labeled examples and ensuring that data to be labeled is selected by the model or a strategy [2,12]. Some classifiers can by combined with belief functions from the theory of Dempster-Shafer [5] in order to take the uncertainty and the imprecision in the labels into account, when the oracle – the person in charge of the labeling task – is not necessarily proficient in the domain.

In this paper, our contribution consists in the use of an evidential variant of the k nearest neighbours classifier which involves belief functions in active learning. More precisely, we compare this evidential classifier to the common k nearest neighbours in a context where the labels provided by the oracle are uncertain.

This article is organised as follow. First of all, we introduce active learning and belief functions in section 2. Our contribution is then described in section 3 and our experiments and results are presented in section 4. Finaly the last section 5 concludes this paper.

2 State of the Art

In this section, we first introduce some notions about active learning (section 2.1) before dealing with the theory of belief functions (section 2.2).

2.1 Active Learning

Active learning (AL) is a subfield of machine learning which allows to limit the amount of training data to train a classifier. Its specificity lies in the construction step by step of a reduced training set, with a limited amount of information [10], by choosing only the most relevant and informative samples which provide an increasing in performances [2,12].

For a given classification problem, let us consider $\mathcal{X} \subset \mathbb{R}^d$, the set of samples described by $d \in \mathbb{N}^*$ features and $\mathcal{Y}$, the set of the different classes. The labeled and unlabeled samples are respectively gathered in $\mathcal{L}$ and $\mathcal{U}$ such that $\mathcal{X} = \mathcal{L} \cup \mathcal{U}$ and $\mathcal{L} \cap \mathcal{U} = \emptyset$. The aim is to label the minimum amount of data required by the model to reach a given performance or the best performance given a budget. The classifier first selects the sample $x \in \mathcal{U}$ whose contribution to the model is supposed to be the most significant before asking for its label to the oracle; this step is called a *query*. Once the label $y \in \mathcal{Y}$ of x is provided, the classifier learns it and updates its knowledge, moving x from $\mathcal{U}$ to $\mathcal{L}$. Queries are formulated repeatedly in the same way until a certain stop criterion is satisfied. At the end of the complete learning process, it is highly probable that $\mathcal{U}$ still contains a lot of samples but it is a matter of little importance as the classifier does not need to learn the whole dataset to perform efficiently.

This strategy is known as the *pool-based sampling* [8] and will be used thereafter.

There are several ways to select and to evaluate the relevance or the "informativeness" of a sample. The utility measures defined by the active learning strategies in the literature [12] differ in their positioning according to a dilemma between the exploitation of the current classifier and the exploration of the training data. Among these strategies, the *uncertainty sampling*, more dedicated to the exploitation part, is one of the most popular and can be divided into three main different types:

- the *least confident* prediction: the sample x^*_{LC} which minimises the probability of classification of its most probable class is queried:

$$x^*_{\mathrm{LC}} = \underset{x \in \mathcal{U}}{\arg\max}\, 1 - P(y_x|x) \qquad (1)$$

With y_x being the most probable class for x according to the classifier. However, this method depends only on the most probable class and do not take the other classes into consideration;
- the *margin sampling*: the sample x^*_{M} which minimises the difference between the probability of classification of the two most probable classes is considered as the most uncertain and will be sent to the oracle:

$$x^*_{\mathrm{M}} = \arg\min_{x \in \mathcal{U}} P(y_x^{(1)}|x) - P(y_x^{(2)}|x) \tag{2}$$

With $y_x^{(1)}$ and $y_x^{(2)}$ being respectively the most and the second most probable classes for $x \in \mathcal{U}$ according to the classifier. The idea is quite natural: if the difference between the probabilities – the margin – is small, then the classification of x would be considered as ambiguous;
- the *entropy sampling*: the sample x^*_{H} which maximises the entropy of Shannon is considered as the most uncertain and will be sent to the oracle:

$$x^*_{\mathrm{H}} = \arg\max_{x \in \mathcal{U}} - \sum_{y \in \mathcal{C}} P(y|x) \log(P(y|x)) \tag{3}$$

Unlike the previous methods, this approach considers every class and not only the most probable ones.

One can also notice that the uncertainty could be "divided" or viewed in two distincts parts [7]: *aleatoric* uncertainty which does not depends on the oracle's knowledge but rather, is inherent to random phenomena – such as tossing a coin; and *epistemic* uncertainty which depends on the oracle's knowledge and ignorance – for example, distinguishing two different species of bird. The second type of uncertainty is more appropriate in our context of active learning as the labeling task strongly relies on the oracle's proficiency in the domain; in particular, it is possible to adapt the uncertainty sampling to epistemic uncertainty [9].

It is however important to notice that the uncertainty described in this section is related to the classifier and not to the oracle. The latter's lack of knowledge and epistemic uncertainty might be modeled with the theory of belief functions introduced in the next section.

2.2 Theory of belief functions

The theory of belief functions also known as theory of Dempster-Shafer is an ideal tool for modeling uncertainty and imprecision [4,13]. Both of these imperfections might be common in labels when the oracle is not an expert in the domain. It is therefore necessary to model these phenomena.

Let us consider a set Θ called the *frame of discernment*, containing the elementary and exclusive hypotheses of a given problem. In the context of classification, it would be the set of classes, therefore: $\Theta = \mathcal{Y}$.

The theory is based on *belief functions* which are defined from 2^Θ to $[0,1]$, with 2^Θ the power set of Θ. The *basic belief assignment* (BBA) $m : 2^\Theta \to [0,1]$ is a belief function which satisfies the *normalisation condition*:

$$\sum_{X \in 2^\Theta} m(X) = 1 \tag{4}$$

A BBA allows to assign elementary belief on different combinations of hypothesis. For exemple, if we consider $X = \{\theta_1, \theta_2, \theta_3\} \in 2^\Theta$, then $m(X)$ is the confidence assigned to θ_1, θ_2 and θ_3 altogether and cannot be subdivided among the different sub-hypothesis $\{\theta_1\}$, $\{\theta_2\}$ or $\{\theta_3\}$; thus, $m(X)$ supports the veracity of X as a whole.

In particular, if $m(\emptyset) = 0$, then the assumption of *closed world* is true, meaning that the frame of discernment is exhaustive. On the contrary, the assumption of *open world* prevails if $m(\emptyset) \neq 0$, which implies that unknown hypothesis exist and may not belong to Θ. In this article, the world will be supposed as closed.

The uncertainty about an hypothesis $\theta \in \Theta$ is modeled by the value of the BBA: the higher the BBA is (close to 1) and the more confidence there is in θ. However, if the value of BBA is low (close to 0), it means that few evidences support θ. The imprecision happens when the oracle hesitates between several hypotheses. This phenomenon is modeled by non empty and non singleton values of 2^Θ and can be extended to the situation of total ignorance, in which case, the entire belief is assigned to Θ in the following way: $m(\Theta) = 1$ and $m(X) = 0$ for all $X \in 2^\Theta - \{\Theta\}$. In a such context, every hypothesis is possible. For example, let us consider a classification problem of 10 classes so that $\Theta = \mathcal{Y} = \{\theta_i \mid i \in [\![1, 10]\!]\}$ and let us note $A = \{\theta_1, \theta_5, \theta_8\} \in 2^\Theta$. If, for a given sample $x \in \mathcal{X}$, the oracle believes that the class of x might be either θ_1, θ_5 or θ_8 without being able to determine which one is the most likely and without having evidences supporting other classes, then a BBA $m_x : 2^\Theta \to [0,1]$ can be defined for x in the following way: $m_x(A) = s$ and $m_x(\Theta) = 1 - s$, with $s \in]0,1[$. Assigning a certain amount of belief $s \neq 1$ in A represents the uncertainty. As A is the union of three different classes, that means that the oracle is furthermore imprecise; he does not particularly favour one answer among the three. Finally, the remaining $1 - s$ of the belief is assigned to Θ, which models the ignorance of the oracle.

The core feature of the Dempster-Shafer's theory is the conjunctive rule of combination of Dempster. It allows the combination of several BBA defined from the same frame of discernment. As a result, the hypotheses on which the BBA agree are enhanced. Let us consider $l \in \mathbb{N}^*$, the BBAs $(m_i)_{i \in [\![1,l]\!]}$ defined on the same frame of discernment Θ will be combined into a single BBA $m_\oplus$:

$$\forall X \in 2^\Theta \qquad m_\oplus(X) = \sum_{X_1 \cap \ldots \cap X_l = X} \prod_{k=1}^{l} m_k(X_k) \tag{5}$$

Even though each BBA m_i respects the assumption of closed world, $m_\oplus$ might not. It is possible to normalise $m_\oplus$ in order to restore the closed world assumption:

$$\forall X \in 2^{\Theta} \qquad m_{\text{Norm}}(X) = \begin{cases} \frac{m_{\oplus}(X)}{1-m_{\oplus}(\emptyset)} & \text{if } X \neq \emptyset, \\ 0 & \text{otherwise.} \end{cases} \tag{6}$$

This rule of combination might be useful in decisions' rules. It is therefore possible to define classifiers based on belief functions as described in the next section.

3 An evidential classifier in active learning

Active learning is a paradigm of machine learning which reduces the amount of training data necessary to train the classifier. As uncertain oracle leads to errors in the labeling task, by coupling AL and the theory of belief functions [15], one can expect a certain robustness towards incorrect labels.

In this section, we present the evidential k nearest neighbours introduced for the first time in [5] (section 3.1) before explaining our approach and its interests in active learning (section 3.2).

3.1 Evidential nearest neighbours

The evidential k nearest neighbours classifier (EkNN) is a variant of the classical k nearest neighbours (kNN) based on belief functions [5].

Let us consider $x \in \mathcal{U}$ and $\tilde{x} \in \mathcal{L}$, one of the k nearest neighbours of x according to the euclidian distance. It is possible to define a BBA $m_{x,\tilde{x}}$ which supports the sole hypothesis that x and $\tilde{x}$ belong to the same class $\theta \in \Theta$. A such BBA must also deal with the distance between x and $\tilde{x}$: the closer they are and the stronger the belief is. Therefore, the BBA might be defined as follow:

$$\forall X \in 2^{\Theta} \qquad m_{x,\tilde{x}}(X) = \begin{cases} \alpha \exp(-\gamma_{\theta} d(x,\tilde{x})) & \text{if } X = \{\theta\}, \\ 1 - \alpha \exp(-\gamma_{\theta} d(x,\tilde{x})) & \text{if } X = \Theta, \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

with α and γ_{θ} parameters and $d(x,\tilde{x})$ the Euclidean distance between x and $\tilde{x}$.

In the original paper [5], it is recommanded to set α to 0.95 and γ_{θ} to the inverse of the mean distance between every training samples from the same class θ.

The combination rule of Dempster is then applied among the BBA which support the same classes. For each class, we get the BBA $(m_{x,(\theta)})_{\theta \in \Theta}$. Finally, the rule of combination is applied one more time among every BBA $(m_{x,(\theta)})_{\theta \in \Theta}$ and we get m_x which aggregates every original BBA supporting different classes.

The decision rule is then the following:

$$\forall x \in \mathcal{U} \qquad y = \underset{\theta \in \Theta}{\arg\max}\, m_x(\{\theta\}) \tag{8}$$

In the context of active learning, it is necessary to take into account the time complexity of the training phase, as the classifier is updated after each query.

In the implementation of EkNN used in this article, the training phase consists to store the training data, then, to compute the γ_θ parameters which take the value of the inverse of the mean distance between every pair of samples from the same class $\theta \in \Theta$. If the euclidian distance is used, then the number of features d will necessarily influence the time complexity. Let us consider $n_\theta \in \mathbb{N}^*$, the number of samples whose class is θ. There are $N_\theta = \frac{n_\theta(n_\theta-1)}{2}$ unique couples of samples, which is also the number of distances to be computed. So the time complexity to compute the mean distance of the class θ is in $O((d+1)N_\theta)$. The global time complexity of the training phase (including every class of Θ) is then in $O\left((d+1)\sum_{\theta\in\Theta} N_\theta\right) = O\left(\frac{d+1}{2}\sum_{\theta\in\Theta}(n_\theta^2 - n_\theta)\right) = O(d\sum_{\theta\in\Theta} n_\theta^2)$. As the training set grows by adding samples, it is not necessary to recalculate the distances that have already been computed; in order to update the mean distance, adding the distances of the new training data and weighting them accordingly is sufficient.

In this context, the fact that the class of an unkown sample x might be one of its neighbour's would be a form of imprecision if there are several different classes. Moreover, the uncertainty could be deduced from the distance: the closer the neighbour is the more believable it is that x belongs to the same class as its neighbour. However, uncertainty and imprecision do not directly depend on the oracle in this situation but only on the training set used by EkNN.

3.2 Use of a belief functions-based classifier in active learning

The interest in using the theory of belief functions is to model uncertainty and imprecision in the data used in active learning. In particular, it becomes possible to model the ignorance, which would be difficult in the classical theory of probability [13]. In the context of crowdsourcing, [1] and [14] applied the theory of belief functions to model the contributors' uncertainty.

The approach described in this article consists to use EkNN in the context of active learning where some labels provided by the oracle are false. Parameters such as level of confidence or expertise estimation can not be taken into account in a such configuration as far as we know. Therefore, the oracle's uncertainty is not modeled in this article. The approach tends rather to limit the influence of erroneous labels given by the oracle. Once the learning phase is over, with some samples being mislabeled, the EkNN classifier uses the density of the distribution and its distances to modulate the influence of each neighbor of a sample to be labeled. It is finally the use of the combination rule (eq. (5)) that will contribute to enhance the hypotheses where the BBAs are in agreement, and therefore, to limit the indirect effect of the oracle's uncertainty.

Let us consider a dataset of two classes $\Theta = \{\theta_1, \theta_2\}$ and the classifier EkNN with $k = 5$. Let x be a sample to be labeled and its 5 nearest neighbours $(x_i)_{i\in[\![1,5]\!]}$ represented in figure 1. Whether the oracle mislabeled the samples (case A) or not (case B), the goal after the learning phase for the classifier is to find the actual class of x. In case B, two neighbours have been mislabeled, therefore, class 1 is majoritary, but the two remaining class 2 (correctly labeled) are closer to x. The classifier will take the distance into account and attribute a greater belief on θ_2 even though the actual class is minoritary in the neighbourhood (see table 2). The coordinates of the samples and their distances to x are described in table 1.

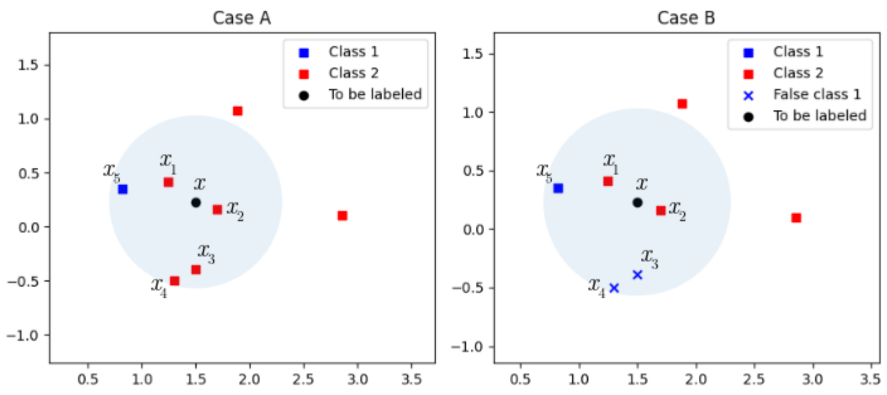


Fig. 1. A sample x to be labeled and its 5 nearest neighbours (within the circle). In case A, every sample has its actual label but in case B, two neighbours have been mislabeled by the oracle.

Table 1. Coordinates and distances to x of its 5 nearest neighbours of figure 1.

Sample	x	x_1	x_2	x_3	x_4	x_5
Coordinate	$(1.5, 0.23)$	$(1.7, 0.15)$	$(1.3, 0.41)$	$(1.5, -0.39)$	$(1.3, -0.51)$	$(0.82, 0.35)$
Distance to x	0	0.22	0.27	0.70	0.62	0.77

Table 2. Values of m_x of figure 1.

	$m_x(\emptyset)$	$m_x(\{\theta_1\})$	$m_x(\{\theta_2\})$	$m_x(\Theta)$
Case A	0	$5.97 \cdot 10^{-3}$	0.919	$7.49 \cdot 10^{-2}$
Case B	0	$5.05 \cdot 10^{-2}$	0.808	0.142

4 Experiments

In this section, we prove that EkNN is a viable classifier in active learning and that its robustness to uncertain labels is interesting. We first present the methodology and protocol used in our experiment (section 4.1) and then discuss and interpret the results (section 4.2).

4.1 Methodology

The EkNN classifier will be compared to kNN in order to highlight the performance of the former in active learning. The value of the nearest neighbours k is difficult to choose. A very small value will make the data very sensitive to noise while a larger value will lead to heavy computations. In this article, the value k will be arbitrarily set to 5 for each experiment and each dataset. These classifiers will also be used in association with uncertainty sampling (least confident prediction) and random sampling to select data for labeling. The secondary sampling method involves selecting the data to be labeled randomly so that its potential contribution or relevance to the model is ignored. Random sampling is often used to highlight a better efficiency of active learning's sampling methods.

Several datasets are used to evaluate the classifiers' behaviour. Synthetic random data have been generated from scikit-learn's library [11] while some real datasets have been extracted from the UCI repository [6]. The different datasets are presented in tables 3 and 4.

Table 3. Synthetic datasets used in the experiments.

Name	**#samples**	**#classes**	**Class distribution**	**#features**
Synthetic A	1,000	2	90 %-10 %	10
Synthetic B	1,000	5	50 %-20 %-15 %-10 %-5 %	10
Synthetic C	1,000	5	75 %-10 %-5 %-5 %-5 %	10

Table 4. Real datasets used in the experiments. In the legend, **LRC** and **MRC** stand for "least represented class" and "most represented class".

Name	**#samples**	**#classes**	**LRC**	**MRC**	**#features**
Speaker Accent Recognition	329	5	8.8 %	50 %	12
HCV	615	5	1.1 %	87 %	14
Letter Recognition (V vs. Y)	1,550	2	49 %	5.5 %	15
Wine Quality (Red Wine)	1,599	6	0.63 %	43 %	11
Pen-Based Recognition of Handwritten Digits (3 vs. 6 vs. 8)	3,166	3	33 %	33 %	16

To evaluate the performance of a classifier coupled to a given sampling method, the *accuracy* criterion is often used, but this metrics is not always relevant, especially when the class distribution is unbalanced [3]. An alternative is to use the *balanced accuracy*, denoted by a_b and defined in the following equation:

$$a_b = \frac{1}{2}\left(\frac{TP}{P} + \frac{TN}{N}\right) \tag{9}$$

With TP, TN, P and N being respectively the amount of true positive, true negative, of samples from the positive class and samples from the negative class.

The results for a combination of a classifier and a sampling method will be plot under the form of a learning curve of balanced accuracy according to the number of queries.

First, a simple comparison between EkNN and kNN is made according to the following protocol. The initial dataset $\mathcal{X}$ is split into a test set $\mathcal{T}$ consisting of 25 % of the samples while a "training" set contains the remaining samples. From the latter, 5 samples of each class is drawn, forming the *bootstrap* $\mathcal{B}$, and will be used to pre-train the classifier. The remaining of the "training" set becomes the *pool* $\mathcal{P}$ so $\mathcal{X} = \mathcal{T} \cup \mathcal{B} \cup \mathcal{P}$. Then, the active learning phase begins: the classifier will forge queries from $\mathcal{P}$ by selecting the sample according to a given sampling method (uncertainty or random sampling) until the number of 150 queries is reached. After each query, the oracle gives the label of the requested sample and the classifier updates its knowledge. Finally, the classifier computes the balanced accuracy on $\mathcal{T}$ and adds it to the learning curve before making another query. The whole process is repeated 20 times such that an averaged learning curve is computed over the 20 learning curves for a given combination of classifier-sampling method.

Second, in order to add some uncertainty to the answers provided by the oracle, noise will be added to the labels, meaning that some of them are replaced by false values. Noise might not always be caused by uncertainty (the oracle can still provide false label while being confident and certain) but it will be sufficient in the current configuration as the uncertainty is studied through its consequences. The protocol used is the same as in the previous experiment except that a copy of the labels is generated with t % of them being noised. To answer queries, the oracle uses this noised copy instead of the original labels. Finally, the noised curves will be compared to the curves corresponding to the data that have not been noised.

4.2 Results

The comparison between EkNN and kNN in figure 2, whether with uncertainty or random sampling, suggests that the contribution of belief functions is most often interesting in AL. It is important to highlight that the EkNN is always superior or equivalent to kNN. This is the case for each dataset except for Speaker Accent Recognition where EkNN's random sampling has significant low performance compared to the other curves. However, the difference is not always significant as the confidence intervals are often overlapping. Due to the definition of the balanced accuracy, the dataset with balanced class distribution presents higher score while the highly unbalanced one generate more learning difficulties. As a sidenote, the fact that uncertainty sampling is superior to random sampling is an expected behaviour, otherwise AL would not be an interesting learning paradigm.

When noise is added to label, it is worth mentioning that on dataset that are easy to classify, the confidence intervals of noised data is wider than non-noised data (Hard). This might be explained by the fact that noised data leads to bigger variance among results, and thus to a less precise balanced accuracy as it can

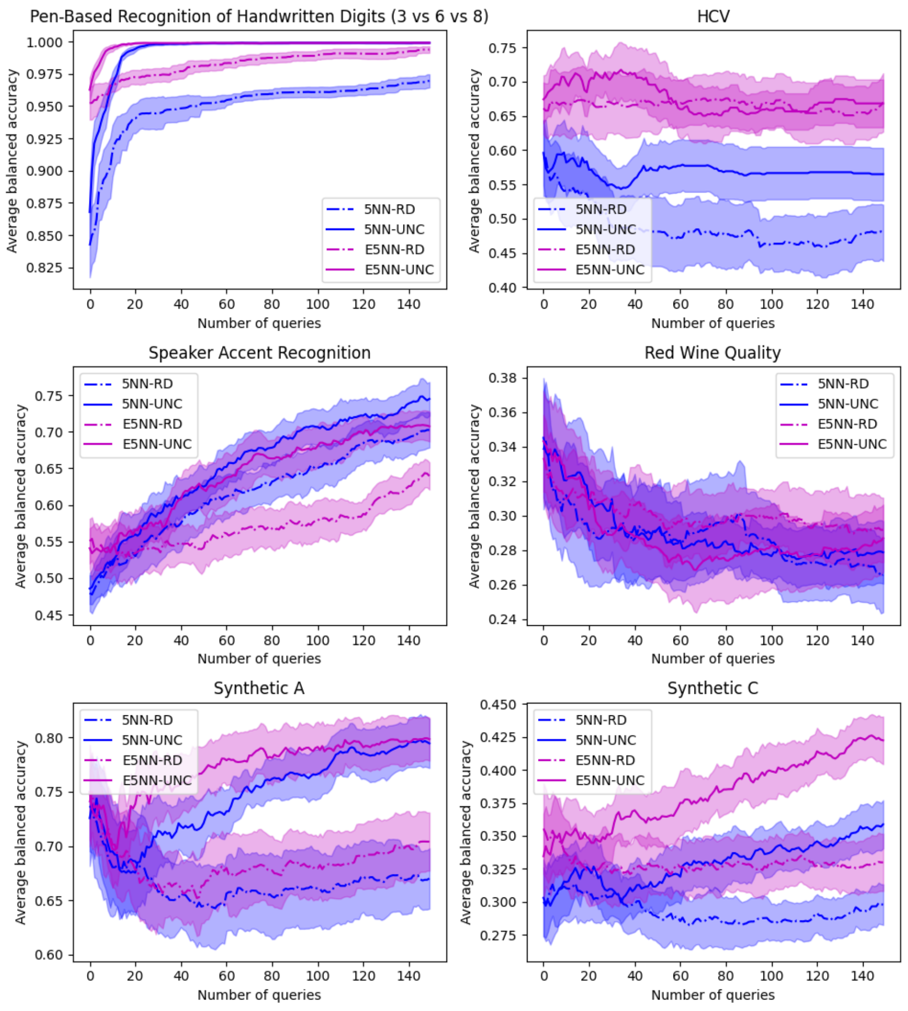


Fig. 2. Comparison of the balanced accuracy (with confidence intervals) between EkNN and kNN through uncertainty sampling (UNC) and random sampling (RD).

be seen in figure 4. Although, this is not always the case as shown in figures 3 and 5. Predictably in such cases, when the noise rate is high, the Hard and Noised curves of a same classifier move away from each other. It is more interesting to compare EkNN and kNN's Noised curves. Again, the former's curve is often above or at the same level as the latter's curve. The gap between the confidence intervals, however, does not appear to be large as they often overlap each other; the gap is wider in figure 5.

Therefore, it appears that EkNN is slightly more robust to noise than kNN.

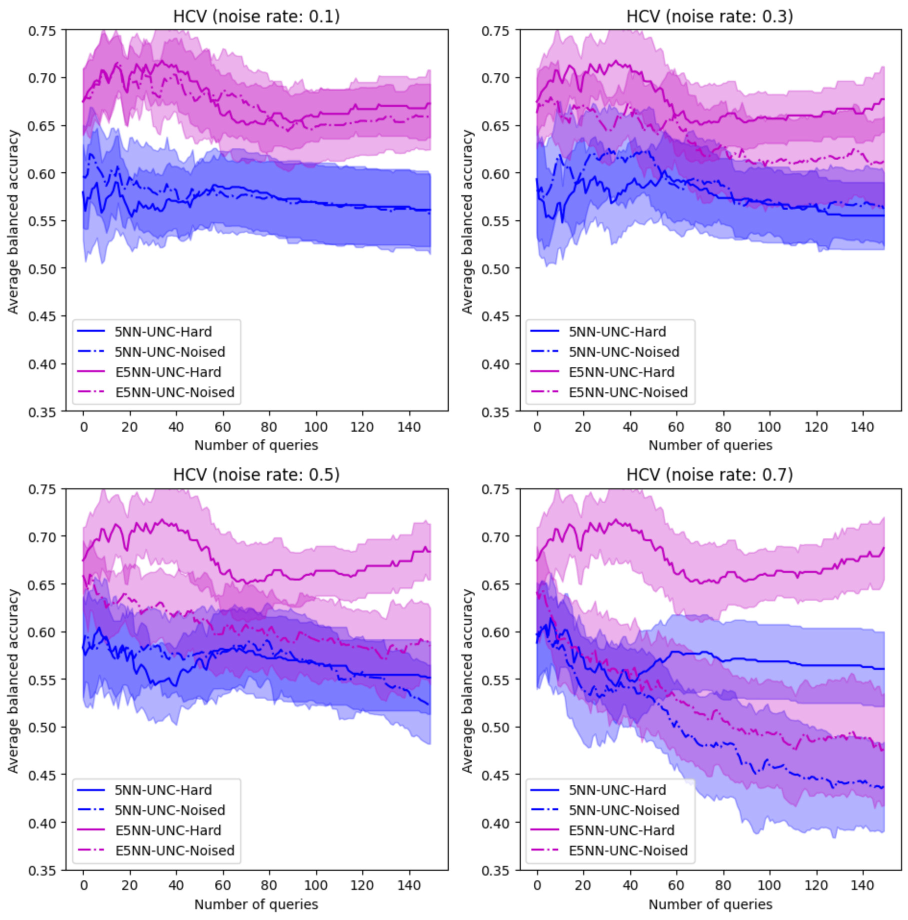


Fig. 3. Comparison of the balanced accuracy (with confidence intervals) between EkNN and kNN on HCV dataset through noised and non-noised (Hard) labels.

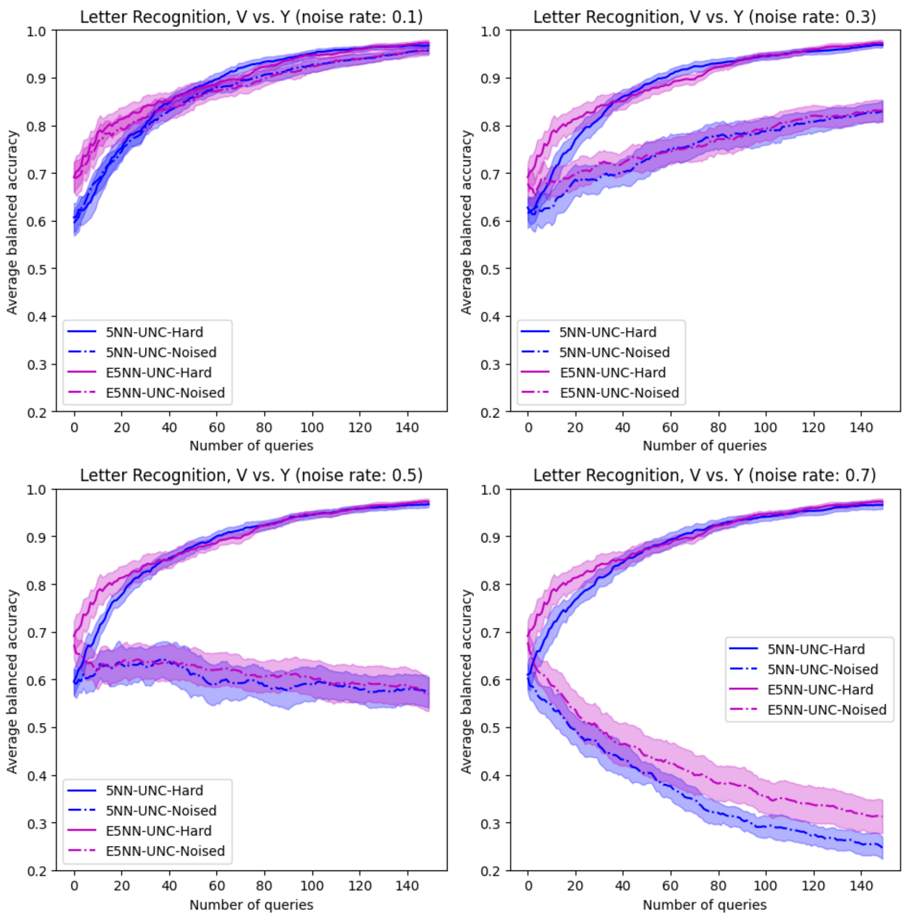


Fig. 4. Comparison of the balanced accuracy (with confidence intervals) between EkNN and kNN on Letter Recognition dataset through noised and non-noised (Hard) labels.

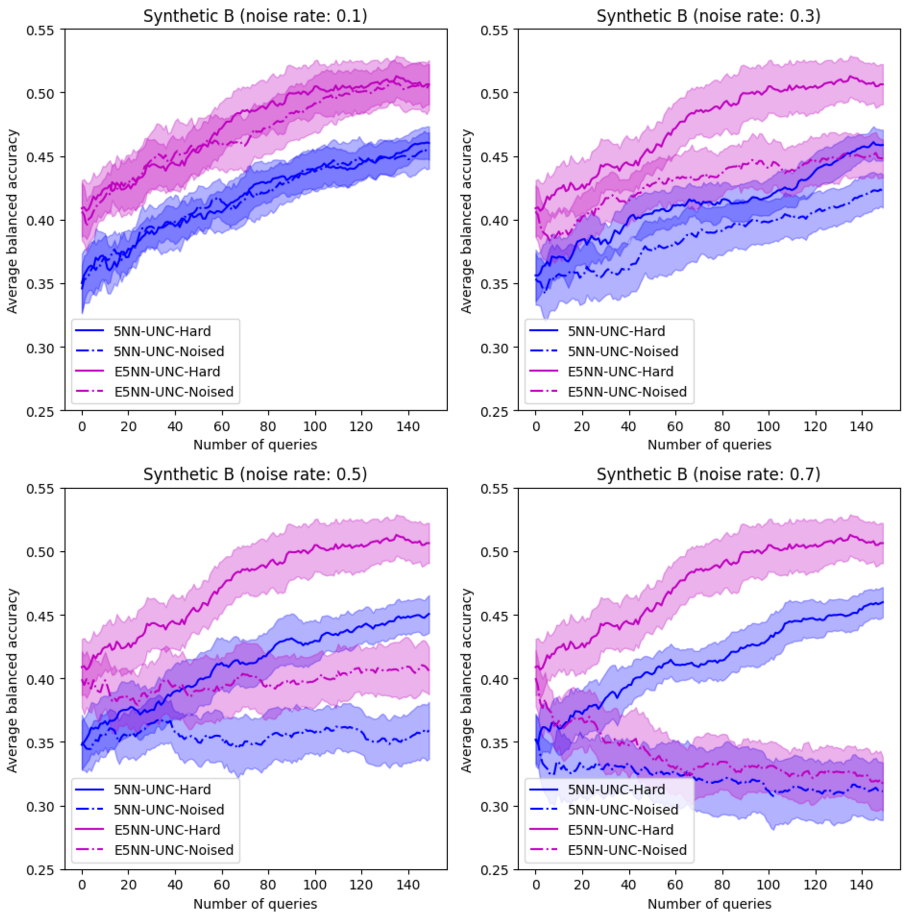


Fig. 5. Comparison of the balanced accuracy (with confidence intervals) between EkNN and kNN on synthetic dataset through noised and non-noised (Hard) labels.

5 Conclusion

Active Learning is a subfield of Machine Learning that aims to reduce the size of the training set and the amount of labels required. This paradigm might be coupled to the theory of belief functions in order to have a way to model uncertainty and imprecision amongst the data.

In this article, we attempt to show the efficiency of a classifier based on belief functions, EkNN, compared to a more common classifier, kNN. Our results suggest that in the majority of the experiments there is a real contribution of the EkNN classifier.

The imperfect label discussed in this article mostly covered the uncertainty aspect with the use of noised label. Further experiments on imprecision could be done in future work to complete this paper. This could be achieved by allowing the oracle to propose several classes instead of one after each query. However, a more flexible classifier would be required to treat imprecision as EkNN is not particularly adapted to deal with several classes per sample. Besides, the EkNN classifier requires heavy computation in AL as the distances between samples are re-computed after each query. Thus, it might be interesting to design a more efficient and adapted belief functions-based classifier for AL in order to treat both uncertainty and imprecision.

References

1. Abassi, L., Boukhris, I.: A worker clustering-based approach of label aggregation under the belief function theory. Applied Intelligence **49**(1), 53–62 (2019). https://doi.org/10.1007/s10489-018-1209-z
2. Bondu, A., Lemaire, V.: État de l'art sur les méthodes statistiques d'apprentissage actif. In: Apprentissage Artificiel et Fouille de Données, AAFD (2006)
3. Brodersen, K.H., Ong, C.S., Stephan, K.E., Buhmann, J.M.: The balanced accuracy and its posterior distribution. In: 2010 20th International Conference on Pattern Recognition. pp. 3121–3124 (2010). https://doi.org/10.1109/ICPR.2010.764
4. Dempster, A.P.: Upper and lower probabilities induced by a multivalued mapping. Ann. Math. Statist. **38**(2), 325–339 (04 1967). https://doi.org/10.1214/aoms/1177698950, `https://doi.org/10.1214/aoms/1177698950`
5. Denoeux, T.: A k-nearest neighbor classification rule based on dempster-shafer theory. IEEE Transactions on Systems, Man, and Cybernetics **25**(5), 804–813 (1995). https://doi.org/10.1109/21.376493
6. Dua, D., Graff, C.: UCI machine learning repository (2017), `http://archive.ics.uci.edu/ml`
7. Hüllermeier, E., Waegeman, W.: Aleatoric and epistemic uncertainty in machine learning: an introduction to concepts and methods. Machine Learning **110**(3), 457–506 (2021). https://doi.org/10.1007/s10994-021-05946-3, `https://doi.org/10.1007/s10994-021-05946-3`
8. Lewis, D.D., Gale, W.A.: A sequential algorithm for training text classifiers. In: Croft, B.W., van Rijsbergen, C.J. (eds.) SIGIR '94. pp. 3–12. Springer London, London (1994)

9. Nguyen, V.L., Destercke, S., Hüllermeier, E.: Epistemic uncertainty sampling. In: Kralj Novak, P., Šmuc, T., Džeroski, S. (eds.) Discovery Science. pp. 72–86. Springer International Publishing, Cham (2019)
10. Nodet, P., Lemaire, V., Bondu, A., Cornuéjols, A., Ouorou, A.: From Weakly Supervised Learning to Biquality Learning: an Introduction. In: In Proceedings of the International Joint Conference on Neural Networks (IJCNN) (2021)
11. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine learning in Python. Journal of Machine Learning Research **12**, 2825–2830 (2011)
12. Settles, B.: Active learning literature survey. Computer Sciences Technical Report 1648, University of Wisconsin-Madison (2009)
13. Shafer, G.: A mathematical theory of evidence, vol. 42. Princeton university press (1976)
14. Thierry, C., Casiez, G., Dubois, J.C., Le Gall, Y., Malacria, S., Martin, A., Pietrzak, T., Uro, P.: Interface de Recueil de Données Imparfaites pour le Crowd-Sourcing. EGC 2020 - Humains et IA, travailler en intelligence Atelier de la conférence (Jan 2020), `https://hal.inria.fr/hal-02465761`
15. Zhu, D., Martin, A., Dubois, J.C., Le Gall, Y., Lemaire, V.: Modèle crédibiliste pour l'échantillonnage en apprentissage actif. In: Rencontres francophones sur la logique floue et ses applications, (LFA) (2021)

SLAYER: A Semi-supervised Learning Approach for Drifting Data Streams under Extreme Verification Latency

Maria Arostegi[1], Jesus L. Lobo[1], and Javier Del Ser[1,2]

[1] TECNALIA, Basque Research and Technology Alliance (BRTA), 48160 Derio, Bizkaia, Spain
[2] University of the Basque Country (UPV/EHU), 48013 Bilbao, Spain
[maria.arostegi,jesus.lopez,javier.delser]@tecnalia.com

Abstract. Classification models learned from data streams often assume the availability of true labels after predicting new examples, either instantly or with some delay with respect to inference time. However, in many real-world scenarios comprising sensors, actuators and robotic swarms, this assumption may not realistically hold, since the supervision of newly classified samples can be unfeasible to achieve in practice. The extreme case where such a supervision is never available is referred to as extreme verification latency. Furthermore, streaming data is also known to undergo the effects of exogenous non-stationary phenomena, by which patterns to be learned from the streams can evolve over time, thereby requiring the adaptation of the classifier for its knowledge to match to the prevailing concept. When these two circumstances (extreme verification latency and concept drift) concur in a given scenario, adapting the model to the evolving dynamics of stream data becomes a challenging task, as the lack of supervision requires rethinking this functionality from a semi-supervised perspective. In this context we present SLAYER, a semi-supervised learning approach capable of tracking the evolution of concepts in the feature space, and analyzing their characteristics towards alleviating the effects of concept drift in the classification accuracy. Besides its continuous adaptation to evolving concepts, another advantage of SLAYER is its resilience against the appearance and disappearance of concepts over time, adapting its knowledge seamlessly when it occurs. We assess the performance of SLAYER over several datasets and compare it to that of state-of-the-art approaches proposed to deal with this stream learning setup. The discussion on the obtained results is conclusive: SLAYER offers a competitive behavior, performing best over several of the datasets considered in the benchmark.

Keywords: Stream learning · extreme verification latency · semi-supervised learning · concept drift

1 Introduction

Nowadays increasing volumes of data are generated at unprecedented speeds, pushing the derivation of new learning models suitable for data analysis under stringent computational constraints. In order to gain value from these data flows, efficient analytical models are needed, which are at the core of research efforts around the Big Data paradigm [13]. Among the technological Big Data landscape, real-time Big Data analytics aim to extract useful information from large datasets, often produced in the form of data streams, namely, sequences of data items that arrive fast and continuously over time. Models that learn from such streaming data while complying with the restrictions imposed by their flowing nature have given rise to a profitable research area widely referred to as data stream learning. Examples of real-world stream learning applications abound in a manifold of domains, including recommendation systems, energy demand modeling, climate data analysis, malware/spam detection, industrial prognosis or traffic data analysis, among many others. As a matter of fact, the upsurge of scenarios resorting to Internet of Things (IoT) devices and functionalities has significantly propelled the necessity of new advances in stream learning, since applications where IoT sensors are deployed often produce huge amounts of data continuously over time [25].

In this context, devices that capture and process such data streams are usually limited in terms of memory and computing power (e.g. sensors, tiny devices), which causes that in most practical cases, the processing time is the main limiting issue to tackle when designing models for stream learning. In response to these restrictions, preprocessing and learning methods have been proposed in the literature, as for instance, selective sampling strategies, divide and conquer strategies and distributed computing [22]. Even if the relative maturity of those developed techniques can satisfy the computational constraints to a certain extent, stream learning models must also be endowed with the capability to adapt their captured knowledge to eventual changes in their received stream data. This phenomenon, known as *concept drift*, is usually due to non-stationary environmental conditions that exogenously affect the production of such data flows, which yields that the characteristics of the data streams evolve over time and impact on the prevalence of the knowledge embedded in the model [11, 12].

The community has hitherto been particularly active in the derivation of new approaches suited to deal with different types of drift as per their speed (abrupt/gradual), severity level or casuistry (feature/label drift). Actually, significant efforts in this vein have been devoted towards the study of concept drift together with known traditional problems in machine learning, such as class imbalance or multi-label classification. However, a scenario that has been less addressed to date is that where concept drift collides with a indefinite lack of supervision of the incoming data. In many practical scenarios annotated stream data can be costly to obtain, or even unfeasible by any means [20]. Therefore, the immediate availability of the supervision associated to stream data (e.g. true labels in classification tasks) cannot be assumed any longer or, at best, supposed to be available after some application-dependent time delay. This circumstance, known in the field as *verification latency*, may hold also in drifting data streams, hence imposing not only efficiency of the learning algorithm, but also a continuous adaptation of the model to varying concepts without any supervision of the input data whatsoever. This need for adaptation is of special relevance in the so-called *extreme* verification latency, in which the supervision of the stream data disappears at a time and is never available again for modeling purposes [18]. The real-world examples provided in the first part of this section also serve as an example where extreme verification latency holds. For instance, IoT sensors that capture temperature or humidity data are often subject to decalibration in the physical and chemical properties of their transducers. In many practical setups companies cannot afford to recalibrate such sensors every time decalibration occurs, nor can they cope with the investment needed to newly annotate data in the new operating regime of the decalibrated sensor. Therefore, they assume that extreme verification latency is an inherent circumstance to be faced by solutions designed for the considered task. Many contributions reported to date around data stream classification over drifting data focus on scenarios characterized by a lagged label supervision with respect to prediction time [17].

This work addresses data stream learning under the above two premises (concept drift and extreme verification latency), focusing on slowly evolving drifts that can be traced and characterized in a non-supervised fashion. To deal with concept drift adaptation in these challenging conditions, we present SLAYER (*Semi-supervised stream LeArning with densitY-basEd dRift adaptation*), a learning algorithm for classification tasks formulated over data streams that can incrementally characterize the evolution over time of the class-dependent modes of streaming data. This characterization relies on an continuous non-supervised analysis of incoming data in order to anticipate changes in their structural characteristics. In other words, the cornerstone of SLAYER is that the analysis is driven by the number of clusters found in data at every time, instead of the number of classes of the formulated classification problem. At this point, we emphasize that, unlike online clustering, which aims to have a good characterization of clusters over time but the correspondence between labels and clusters is not taken into account, in our setup it is crucial to trace the correspondence between labels and concepts over time. Therefore, the goal is to predict examples to classes rather than to infer how data organize in clusters over time [21].

We assess the performance of the proposed approach over a set of public synthetic datasets featuring evolving drifts of very diverse nature, assuming extreme verification latency in all of them. Results of our proposed scheme are compared to the ones obtained by other methods reported in the scarce literature that has so far undertaken this same problem. Results elucidate that the unsupervised drift tracking mechanisms embodied in SLAYER can lead to superior accuracies than its counterparts, as they allow for a fine-grained modeling of the information continuously flowing and drifting over time.

The rest of the paper is structured as follows: first a brief review of related contributions is made in Section 2. Next, Section 3 describes the overall algorithmic steps of SLAYER, jointly with a description and design rationale of methods underneath. The experimental setup is detailed in Section 4, whereas results are presented and discussed in Section 5. Finally, Section 6 concludes this work and outlines future research directions stimulated by our findings.

2 Background

Among the extensive bibliography corpus related to stream learning, we first pause at the conclusions drawn in [15] and [16]. The importance of dealing with verification latency when learning from streams was already pointed out in the former, whereas the latter reviewed more than 130 works about concept drift, analyzing assumptions, methodologies and techniques, and concluding with prospects and guidelines for future research in the field. Among them, extreme verification latency was identified as an area deserving further research. The delayed or null supervision of incoming data samples implies the use of different strategies that blindly monitor the distribution of arriving data samples and adapt the model to changes detected therein [17]. Within such strategies to handle verification latency, we highlight 1) semi-supervised learning [12], in which a few labeled samples are available for initially training a model, which allows subsequently extracting further knowledge from the large amount of unsupervised streaming data; and 2) active learning [19], by which the learning method itself chooses the instances to be learned. Next, we review the main algorithms contributed so far for data stream classification in non-stationary environments subject to extreme verification latency.

To begin with, the Arbitrary Sub-Populations Tracker (APT) approach proposed in [14] is characterized by a two-stage learning strategy. Expectation-maximization is used to determine the optimal one-to-one assignment between the unlabeled and labeled data (by using kernel density estimation techniques), and next the classifier is updated to reflect the population parameters of newly received data, as well as the drift characteristics. Shortly thereafter, the renowned Compacted Object Sample Extraction (COMPOSE) semi-supervised approach was reported in [5], which is essentially a geometry-based framework capable of learning from non-stationary streaming data by following three steps: 1) the extraction of so-called α-shapes to represent the current class conditional distribution; 2) the shrinkage of such α-shapes to properly model the geometric center of each class distribution; and 3) from the compacted α-shapes new instances are extracted to serve as labeled data for future time steps.

Ever since COMPOSE has originated multiple variants, such as FAST COMPOSE, MASS, or LEVELIW, which improve the original version of the algorithm in different aspects (e.g. speed). Before commenting on them, we follow our review by mentioning the Classification Algorithm Guided by Clustering (SCARGC) in [20], which consists of clustering followed by classification applied repeatedly in a closed-loop fashion. The algorithm exploits the current and past cluster positions extracted from unlabeled data to track drifts over time. Later in time, the first improvement of COMPOSE, denoted as FAST COMPOSE, was published in [23], which reduces the computational fingerprint of COMPOSE by alleviating the complexity of their shape extraction step. At the time, another modified version of COMPOSE – Modular Adaptive Sensor System (MASS) [8] – was proposed as a workaround to extreme verification latency in stream data, yet was still found to be computationally unaffordable for resource constrained applications (e.g. IoT sensor networks). Likewise, LEVELIW [23, 24] was proposed as a framework for learning in extreme verification latency scenarios by using importance weighting in gradual concept drift scenarios. LEVELIW leverages weighting to match distributions between two consecutive time steps, and estimates the posterior distribution of the unlabeled data using a weighted least-squares probabilistic classifier. The work in [1] proposed TRACE, a technique that predicts the trajectory of concepts in the feature space by means of Kalman filtering, which was shown to adapt to drifting environments without any external supervision. Finally, [7], a semi-supervised density-based adaptive model for non-stationary data (AMANDA) has been recently proposed in [7]. AMANDA weights and filters samples that best represent the concepts in the distribution. To this end, it identifies which instances lie in the core region of the existing class distributions, so that these selected instances are chosen as training data for the next iteration. The weighting method receives a set of instances as input and returns the same set of instances associated to weights.

In conclusion, the recent activity in the field reviewed above suggests that this is a topic eager for new algorithmic approaches. This is the main motivation for the development of SLAYER, which incorporates a novel perspective to predict the evolution of drifts over stream data in an unassisted manner.

3 SLAYER: Description and Design Rationale

Learning in non-stationary streaming environments can be approached from two different perspectives: active or passive [6]. The difference among them resides on the adaptation mechanism: the active procedure depends on the use of a drift detector that processes arriving data and triggers an alarm when a change is detected. By contrast, a passive method continuously modifies the model over time without explicitly detecting the drift in order to prepare the model for any concept drift eventually present in the stream data. Our scheme builds upon a passive approach: even if a drift is not detected anyhow, SLAYER constantly updates its knowledge in order to yield a prediction conforming to the prevalent status of the stream. For this to occur, a fundamental premise of SLAYER is that drifts over the streams are gradual, so that changes can be monitored and tracked over non-supervised data. This is important to be noted, as conventional drift detectors and passive adaptation strategies operate by assuming immediate access to the annotation of the received data instances, so that changes can be inferred by quantifying the performance degradation of the model over time.

Figure 1 and Algorithm 1 summarize the main steps of SLAYER. As can be read in the algorithm, we departing from the arrival of the initial set of labeled instances $\{\mathbf{x}_t\}_{t=1}^{T_0}$, from which SLAYER infers the clusters in which these initial samples can be grouped (line 2). Without loss of generality, this initial step can be done by very assorted means. For the sake of simplicity and given the limited computational effort imposed by streaming setups, this is done by using K-means together with the well-known elbow method to compute the optimal number of clusters K_0: the total within-cluster variation is calculated for increasing values of K_0 as the sum of squared distances Euclidean distances between items and the corresponding centroid, so that the optimal value of K_0 is declared when the addition of a new cluster does not imply an improvement (decrease) of the variation measure.

Once clusters $\{\mathcal{X}_0^k\}_{k=1}^{K_0}$ have been computed over the initial set of samples $\{\mathbf{x}_t\}_{t=0}^{T_0}$, two structural characteristics are quantified for every cluster $\mathcal{X}_0^k$: the amount Q_0^k of examples assigned to the cluster, i.e. $Q_0^k = |\{\mathbf{x}_t \ \forall t \leq T_0 : \texttt{cluster}(\mathbf{x}_t) = k\}|$, and its *radius* R_0^k (namely, the maximum distance from every example to the centroid of its assigned cluster). From these characteristics and assuming a globular shape of the cluster, a rough estimation of the cluster density is given by $D_0^k = Q_0^k/(R_0^k)^D$, where D is the dimension of the feature space (line 2). After computing this measure over the cluster space of the initial batch of labeled examples, a 1-nearest neighbor model can be used over such samples to predict the samples of the next batch that follows immediately thereafter (Lines 3 and 4).

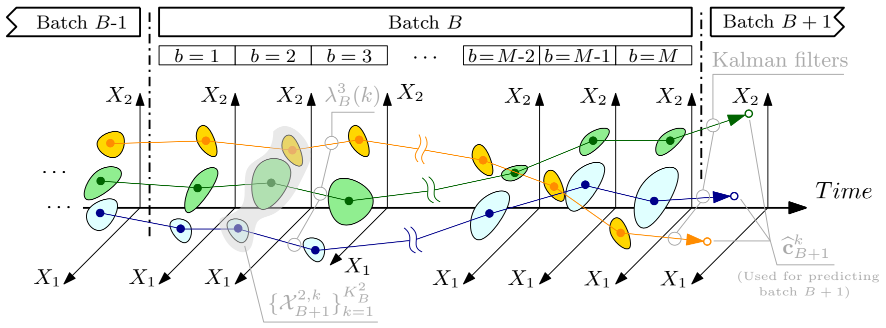


Fig. 1: Schematic diagram illustrating the internal processing of unsupervised batches featured by SLAYER, including the successive clustering and mapping procedure performed at every mini-batch $b \in \{1, \ldots, M\}$.

When the next batch arrives and once samples therein have been predicted (line 6), another clustering phase is performed over such samples (line 7) so that the new cluster space $\{\mathcal{X}_1^k\}_{k=1}^{K_1}$ can reflect the emergence of new clusters or the disappearance of others. A matching between the clusters from the previous batch and those arising for the current one is done (line 8), yielding a mapping function $\Omega_1 : \{1, \ldots, K_1\} \mapsto \{1, \ldots, K_0\}$ given by:

$$\Omega_1(k') = \underset{k \in \{1,\ldots,K\}}{\arg\min} \|\mathbf{c}_1^{k'} - \mathbf{c}_0^k\|_F, \tag{1}$$

where $\|\cdot\|_F$ stands for Frobenius (Euclidean) norm, and $\mathbf{c}_1^k$ denotes the centroid of the k-th cluster. Despite its simplicity, this simple mapping rule permits to trace how the cluster space evolves between subsequently received batches. Departing from this matching, densities $\{D_1^k\}_{k=1}^{K_1}$ of the K_1 clusters are computed and compared to those of the previous clusters to which they are mapped. SLAYER implements this comparison as $|D_1^{k'} - D_0^{\Omega(k')}|$, which, together with an increase or decrease of the number of clusters (namely, when $K_1 \neq K_0$), is an

Algorithm 1: SLAYER

```
Input : Initially annotated data instances $\{\mathbf{x}_t, y_t\}_{t=1}^{T_0}$, threshold $\epsilon$ for
        declaring a density-driven change, number of mini-batches $M$,
        unsupervised stream instances $\{\mathbf{x}_t\}_{t>T_0}$.
Output: Predictions $\{\widehat{y}_t\}_{t=1}^{\infty}$.
1  Let $L$ denote the number of classes
2  Compute clusters $\{\mathcal{X}_0^k\}_{k=1}^{K_0}$ and densities $\{D_0^k\}_{k=1}^{K_0}$ for $\{\mathbf{x}_t\}_{t=1}^{T_0}$
3  Set $\widehat{\mathbf{c}}_1^k$ equal to the average of all $\mathbf{x}_t$ such that $\mathbf{x}_t \in \mathcal{X}_0^k$
4  Set $\ell(k)$ to the majority class of annotated instances assigned to cluster $k$
5  foreach incoming batch $B \in [1, \ldots, \infty]$ do
6      Predict samples $y_t \in B$ as the label $\ell(k)$ of the closest $\{\widehat{\mathbf{c}}_B^k\}_{k=1}^{K_{B-1}}$
7      Compute clusters $\{\mathcal{X}_B^k\}_{k=1}^{K_B}$ and densities $\{D_B^k\}_{k=1}^{K_B}$ for $\{\mathbf{x}_t\}_{t\in B}$
8      Compute mapping $\Omega_B()$ between $\{\mathcal{X}_B^k\}_{k=1}^{K_B}$ and $\{\mathcal{X}_{B-1}^k\}_{k=1}^{K_{B-1}}$
9      Divide batch $B$ in $M$ mini-batches
10     foreach minibatch $b$ do
11         Compute clusters $\{\mathcal{X}_B^{b,k}\}_{k=1}^{K_B^b}$ and centroids $\{\mathbf{c}_B^{b,k}\}_{k=1}^{K_B^b}$
12         Infer mapping $\lambda_B^b : \{1, \ldots, K_B^b\} \mapsto \{1, \ldots, K_B^{b-1}\}$
13         Correct each $\mathbf{c}_B^{b,k}$ based on $\lambda_B^b(\cdot)$ as per Expr. (2)
14         if $\exists k$ such that $|D_B^k - D_B^{\Omega_B(k)}| > \epsilon$ then
15             Set $\alpha_\ell = 0 \ \forall \ell \in \{1, \ldots, L\}$
16         else
17             Adjust $\alpha_\ell$ for each $\ell \in \{1, \ldots, L\}$ as in Expression (3)
18         end
19         Predict $\widehat{\mathbf{c}}_{B+1}^k$ by means of a Kalman filter, using as prior estimation
            $\widehat{\mathbf{c}}_B^{k^*}$, where $k^* = \lambda_B^1(\lambda_B^2(\ldots(\lambda_B^{M-1}(\lambda_B^M(k)))\ldots)$
20     end
21 end
```

indicator of a potential change of the cluster space over time. For this purpose, a change is declared when there exists at least a cluster $k' \in \{1, \ldots, K_1\}$ for which $|D_1^{k'} - D_0^{\Omega_1(k')}| > \epsilon$, where ϵ is an hyper-parameter of SLAYER that tunes the sensitivity of the model to the speed and intensity at which clusters vary.

When a change is identified, a forgetting mechanism must be triggered to discard previous knowledge about the stream. In the unsupervised regime, SLAYER predicts arriving samples by assigning them the label associated to the prevailing cluster space whose centroid is closest to each sample. To this end, SLAYER attains a finer level of granularity by dividing each batch into M *mini-batches* of equal size (line 9). Samples falling inside each mini-batch are clustered by means of a memory-based K-means algorithm wherein, once K_1^b clusters have been extracted from mini-batch b (line 11), a distance-based matching $\lambda_1^b : \{1, \ldots, K_1^b\} \mapsto \{1, \ldots, K_1^{b-1}\}$ is computed as in (1) (line 12), so that centroids can be corrected as (line 13):

$$\mathbf{c}_1^{b,k} = \frac{\alpha_{\ell(k)} \cdot \mathbf{c}_1^{b-1,\lambda_1^b(k)} \cdot N_1^{b-1,\lambda_1^b(k)} + (1-\alpha_{\ell(k)}) \cdot \mathbf{c}_1^{b,k} \cdot N_1^{b,k}}{\alpha_{\ell(k)} \cdot N_1^{b-1,\lambda_1^b(k)} + (1-\alpha_{\ell(k)}) \cdot N_1^{b,k}}, \tag{2}$$

where $N_B^{b,k}$ denotes the number of samples assigned to cluster k in mini-batch b of batch B, $\mathbf{c}_B^{b,k}$ its centroid, and $\alpha_\ell(k)$ is a forgetting factor that depends on $\ell(k)$, i.e. the label of cluster k that is *tied* through consecutive batches by virtue of repeated clustering and mapping over mini-batches. In the above expression we note that $\alpha_{\ell(k)} = 0$ implies no correction of the cluster center, whereas $\alpha_{\ell(k)} = 1$ denotes full persistence of the cluster space over the batch. Given this role of $\alpha_{\ell(k)}$ in the update dynamics of the cluster space, SLAYER modifies its value whenever a new batch arrives: if a change is declared, $\alpha_\ell = 0$ for all classes (line 15). Otherwise (line 17), α_ℓ is set inversely proportional to the average distance between centroids of tied clusters that are assigned label ℓ, i.e.:

$$\alpha_\ell \propto \left(\sum_{k=1}^{K_1^b} \|\mathbf{c}_1^{b,k} - \mathbf{c}_1^{b-1,\lambda_1^b(k)}\|_F \cdot \mathtt{I}(\ell(k) = \ell) \right)^{-1}, \tag{3}$$

where $\mathtt{I}(\cdot)$ is an auxiliary binary function taking value 1 if its argument is true (0 otherwise). By updating the forgetting factor as in the above expression, α_ℓ can be thought to be a rough estimation of the average *speed* at which label concepts move over the feature space during the batch. The above adaptation of α is done in a per label basis and not in a finer granularity (per every cluster), as this could increase significantly the overall computational complexity of SLAYER, specially in those cases with scattered cluster spaces.

Finally, SLAYER attains a higher level of adaptability against drifts over the stream by *predicting* where clusters will reside during the incoming batch (line 19). Since there is a correspondence (thoroughly tied through mini-batches) between the clusters discovered in consecutive batches, a lightweight Kalman filter is used to estimate the future coordinates $\widehat{\mathbf{c}}_{B+1}^k$ of every centroid at the end of the current batch B. A Kalman filter is a simple recursive system used to calculate the state of a linear dynamic system and the variance or uncertainty of the estimate. In SLAYER, a Kalman filter keeps track of the estimated position of cluster centers in the feature space, yielding a vector of estimated centroids $\{\widehat{\mathbf{c}}_{B+1}^k\}$ that is used for predicting the next batch of samples. This is possible thanks to their assigned labels propagated through the mini-batch-wise cluster-and-mapping process explained above.

4 Experimental Setup

In order to assess the performance of SLAYER, we make use of a public repository of non-stationary data streams widely adopted by the stream mining community working with gradual drifts [20]. Specifically, the repository contains 15 synthetic datasets featuring different drift changes over time, including translations, rotation, warps and other transformations of the feature space. Table 1 summarizes their main characteristics. The column labeled as D and L refer to the number of features and classes, respectively. In these datasets, the initial 5% of the samples are assumed to be supervised, whereas the remaining stream instances arrive in 100 batches in chronological order.

After reviewing the latest contributions on data stream classification under extreme verification latency in non-stationary environments (Section 2), we have built a comparison benchmark that includes several proposals published so far in this research area:

- A static classifier learned from the first labeled samples (STATIC).
- A sliding window classifier that learns initially from labeled samples, and updates its knowledge with predicted upcoming samples, discarding samples that do not fall inside the sliding window for predicting new instances (SLIDING).
- An incremental window classifier, which works similarly to the sliding window classifier, but that does not forget any past instance for training (INCR).
- COMPOSE [5], which creates a boundary from the current data and defines a shape that represents the distribution of each class. After several iterations, COMPOSE draws instances from the core region(s) to support training as labeled data. Finally, upon the reception of new unlabeled data, new instances are combined with that of the core regions to retrain the model and adapt to eventual non-stationary behaviors over the stream.
- LEVELIW [23, 24], an iterative weighting approach that relies on the assumption that there is an overlap among class-conditional distributions between consecutive time steps, a premise that holds in slow drifting data streams.
- AMANDA [7], in which the distributions of each class are first estimated over the received labeled samples. Then, a semisupervised learning classifier is learned and used to predict upcoming batches with unlabeled samples. A density-based algorithm measures the importance of the classified instances by weighting them, and only retaining the most representative samples. This method has two variations: AMANDA-FCP (A-FCP), which selects a fixed number of samples; and AMANDA-DCP (A-DCP), which dynamically selects samples from data.

Dataset	**L**	**D**	**Instances**	**Class distribution**	**Description**
`1CDT`	2	2	$16 \cdot 10^3$	50%/50%	Two clusters (one per class), one moving in diagonal
`1CHT`	2	2	$16 \cdot 10^3$	50%/50%	Two clusters (one per class), one moving horizontally
`1CSurr`	2	2	55283	37%/63%	Two clusters (one per class), one surrounding the other
`2CDT`	2	2	$16 \cdot 10^3$	50%/50%	Two clusters (one per class), moving in the same diagonal
`2CHT`	2	2	$16 \cdot 10^3$	50%/50%	Two clusters (one per class), moving together horizontally
`FG2C2D`	2	2	$2 \cdot 10^5$	75%/25%	Three clusters for one class, one moving cluster for the other class
`GEARS`	2	2	$2 \cdot 10^5$	50%/50%	Two rotating gears, one per class
`MG2C2D`	2	2	$2 \cdot 10^5$	50%/50%	Two clusters per class, moving and overlapping with each other
`UG2C2D`	2	2	$1 \cdot 10^5$	50%/50%	One cluster per class, moving without overlapping with each other
`UG2C5D`	2	5	$2 \cdot 10^5$	50%/50%	Two 5-dimensional clusters, one per class, moving and overlapping
`4CE1CF`	5	2	173250	20% per label	Four classes expanding and one class fixed, one cluster each
`4CR`	4	2	144400	25% per label	Four clusters, one per class, rotating with no overlap
`4CRE-V1`	4	2	$125 \cdot 10^3$	25% per label	Four clusters, one per class, rotating with expansion (version 1)
`4CRE-V2`	4	2	$183 \cdot 10^3$	25% per label	Four clusters, one per class, rotating with expansion (version 2)
`5CVT`	5	2	$4 \cdot 10^4$	33%×1, 16%×4	Five clusters, one per class, moving together vertically

Table 1: Slow drifting stream datasets from [20] utilized in this work.

In what refers to performance metrics, we use the so-called *prequential error*, which has been thoroughly used in the literature [9, 10]. For the sake of compliance with the methodological practices in the above prior work, the prequential error is computed based on an accumulated sum of a loss function between the prediction and observed values [3], i.e.:

$$preqError(t) = \frac{1}{t}\sum_{t'=1}^{t} \mathcal{L}(y_{t'}, \widehat{y}_{t'}), \tag{4}$$

where the prequential error is computed at time t, $\widehat{y}_{t'}$ represents the prediction, and $y_{t'}$ represents the real value at time $t' \leq t$. Among the possible loss functions for classification, we specifically use the 0-1 loss, i.e. $\mathcal{L}(y_{t'}, \widehat{y}_{t'}) = 0$ if $y_{t'} = \widehat{y}_{t'}$ and 1 otherwise. The prequential error allows monitoring the evolution of the models' performance over time. However, it is convenient to gauge other performance metrics that are sensitive to mild class imbalance, as there is no certainty that classes are equally represented in batches received over time. Therefore, we also report on the macro F1 score, which grants the same importance to each label/class. Source code and datasets will be made available at a public repository available in https://git.code.tecnalia.com/maria.arostegi/slayer/.

5 Results and Discussion

The obtained results are summarized in Tables 2 (average prequential error) and 3 (average macro F1 score) for the methods and datasets considered in our study. In these tables, the best results for each datasets are highlighted in bold.

Dataset	STATIC	SLIDING	INCR	COMPOSE	LEVELIW	A-FCP	A-DCP	SLAYER
1CDT	0.76	0.06	0.3	0.08	0.04	0.02	0.05	**0.006**
1CHT	3.93	0.43	3.2	0.48	0.4	**0.33**	0.39	0.38
1CSURR	35.86	9.05	36.06	9.43	9.2	**4.39**	7.93	5.91
2CDT	46.3	6.13	46.14	6.73	49.74	5.46	5.83	**3.8**
2CHT	45.97	48.45	46.01	47.41	47.41	14.39	19.93	**10.27**
FG2C2D	17.79	4.43	18.29	12.15	4.31	5.12	16.39	4.32
GEARS	5.43	0.81	5.33	4.03	6.18	**0.81**	3.74	3.84
MG2C2D	51.63	22.86	50.66	49.17	9.31	8.7	14.88	**8.59**
UG2C2D	55.81	4.97	54.42	5.32	26.34	4.3	12.64	**4.26**
UG2C5D	30.97	20.11	30.62	20.82	20.82	8.21	8.53	**8.08**
4CE1CF	1.98	1.9	1.82	2.09	2.21	**1.73**	1.92	2.029
4CR	78.83	0.02	78.75	0.04	0.02	0.02	0.03	**0.009**
4CRE-V1	78.15	81.29	79.44	79.55	79	73.5	73.28	**2.21**
4CRE-V2	79.61	82.88	79.67	77.38	80.77	69.97	71.81	**7.69**
5CVT	54.51	60.97	52.04	65.5	59.18	24.11	52.38	**12.4**

Table 2: Prequential error of the compared methods for the considered datasets.

We begin our discussion by inspecting the prequential error scores in Table 2. It is first relevant to notice that the use of naïve methods (STATIC, SLIDING, INCR) yields in general comparatively bad results due to the fact that they are not designed to cope with non-stationary environments under extreme verification latency. If we take a closer look at these scores, except for GEARS, 4CR and 4CEF1CF, the prequential error of those methods are significantly higher that the rest of algorithms in the benchmark. By contrast, if we consider approaches tailored to deal with unsupervised drifting streams such as COMPOSE or LEVELIW, results improve across datasets, yet still lagging behind those obtained by A-FCP, A-DCP and SLAYER.

Among the weak points of our proposed approach we pause at the importance of the density of the clusters, and the speed at which they move over the feature space. For example, in the 4CE1CF dataset, in the beginning of the stream the 5 classes are very close to each other in the feature space, represented by single clusters with very similar densities. This poses a challenge for the change detection mechanism of SLAYER, as reflected by the 6th position in the ranking between models for this dataset. Therefore, SLAYER fails to properly characterize the evolution of concepts from unsupervised data streams when these two circumstances collide together.

We now focus on the comparison between the two AMANDA-based approaches (A-FCP and A-DCP) and SLAYER. First we observe that SLAYER obtains a slight advantage over A-FCP, as SLAYER scores best in 10 out of the 15 datasets under consideration. A-FCP attains the best performance in 4 datasets, falling down once to the 6^{th} position in the ranking. By contrast, A-DCP is never below the fourth position among the methods, but it does not score best in any of the datasets. Similar conclusions can be drawn when analyzing the macro F1 results shown in Table 3. SLAYER yields the best results for 9 datasets, followed by A-FCP that scores first in 5 datasets, and A-DCP in just one dataset.

Dataset	STATIC	SLIDING	INCR	COMPOSE	LEVELIW	A-FCP	A-DCP	SLAYER
1CDT	0.9935	0.9994	0.9971	0.9995	0.9996	0.9997	0.9994	**0.9999**
1CHT	0.96	0.995	0.9681	0.9949	0.996	**0.9963**	0.9955	0.996
1CSURR	0.6403	0.9137	0.6384	0.9094	0.6368	**0.9607**	0.9267	0.9385
2CDT	0.3871	0.9418	0.3884	0.9362	0.4836	0.948	0.9416	**0.961**
2CHT	0.3954	0.356	0.3942	0.4758	0.4758	0.8526	0.788	**0.897**
FG2C2D	0.7322	0.9391	0.7298	0.8596	**0.9469**	0.9319	0.819	0.9419
GEARS	0.9474	0.9957	0.9485	0.9637	0.9382	**0.9957**	0.963	0.9579
MG2C2D	0.4795	0.7543	0.4936	0.505	0.5923	**0.9143**	0.8499	0.9133
UG2C2D	0.4425	0.9514	0.4546	0.9491	0.7366	**0.9581**	0.8706	0.9538
UG2C5D	0.668	0.7549	0.6782	0.7918	0.7918	0.9151	0.9129	**0.9153**
4CE1CF	0.9807	0.9795	**0.9821**	0.9781	0.9779	0.9808	0.9803	0.9798
4CR	0.2099	0.9998	0.2154	**0.9999**	0.9998	0.9998	**0.9999**	**0.9999**
4CRE-V1	0.2073	0.1804	0.1997	0.2035	0.2486	0.267	0.2651	**0.9778**
4CRE-V2	0.2043	0.1259	0.2039	0.1971	0.2464	0.3035	0.181	**0.9229**
5CVT	0.3537	0.1812	0.3707	0.2385	0.1767	0.7297	0.3802	**0.8849**

Table 3: Macro F1 results of the compared methods for the considered datasets.

We end our discussion by inspecting the statistical significance of the differences observed in the above tables. To shed light on this matter, Demsar's critical distance diagrams are often used [4]. These diagrams show the average ranks of a number of models under comparison across multiple datasets, wherein significant differences are declared when the difference between the average ranks of two models is larger than a critical distance (CD). The CD value is given by a post-hoc Nemenyi test at a certain confidence level (usually set to 0.95). The critical distance diagram corresponding to the results in Table 2 is shown in Figure 2.a: unfortunately, the average rank achieved by SLAYER cannot be declared to be statistically significant, mainly due to the relatively high value of CD (2.71) that results from the large number of models being compared.

Although critical distance diagrams are widely used by the community, they have been reported to be misleading and scarcely interpretable [2]. Therefore, we complement this analysis with a pairwise Bayesian analysis of differences between the best performing methods in the benchmark, namely, A-FCP, A-DCP and SLAYER. Specifically, we model the probability that one model outperforms another based on the results obtained by each of them over all datasets. Once fitted, the probability distribution is sampled and displayed in barycentric coordinates, wherein three regions can be observed: the first algorithm outperforms the second (and vice versa), and a region of practical equivalence (the two methods perform similarly). A *rope* parameter establishes the minimum difference that scores of both methods must have for them to be declared different to each other, thereby ensuring that statistical significance relies on an interpretable parameter. Figure 2.b depicts the Bayesian posterior plot between A-FCP and SLAYER performed over the prequential error scores with a rope equal to 0.1. As shown in this plot, the sampled distribution appears to be clearly skewed towards SLAYER, revealing that it is likely that our proposal outperforms A-FCP, with prequential error differences larger than the selected *rope* value. When comparing SLAYER to A-DCP (Figure 2.c), the significance of the better prequential error performance observed for our proposal is even more significant.

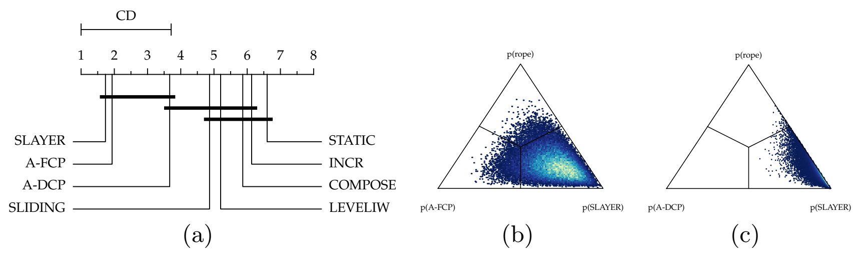


Fig. 2: (a) Critical distance diagram of prequential error results in Table 2; (b) Bayesian posterior plot of A-FCP versus SLAYER corresponding to the same table; (c) Bayesian posterior plot of A-DCP versus SLAYER.

6 Conclusions and Future Work

This work has gravitated on a research area that has been paid considerably lower attention in the stream learning field than other widely studied paradigms: learning to classify streams subject to the effects of concept drift and extreme verification latency. In this setup, classification models for streaming data become obsolete at a point due to the drift experienced by concepts to be classified over time, whereas the absence of supervision about the stream samples hinders severely the change detection, tracking and adaptation processes. This confluence of conditions has been the main motivation for SLAYER, the novel approach presented in this paper, which continuously analyzes the evolution of clusters and *ties* them together across batches, so that a correspondence between labels and evolving clusters can be constantly maintained over time. By virtue of this mechanism, SLAYER seamlessly accommodates the appearance/disappearance of new clusters, and is particularly suitable for environments where the speed of change in the feature space is not abrupt, so that posterior distributions overlap to a certain extent between consecutive time ticks (*slow* feature drift). Simulation results over 15 datasets with different drift dynamics have elucidated that in most of them, SLAYER outperforms state-of-the-art approaches contributed to address this kind of scenarios. Furthermore, differences have been found to be significant as concluded from a Bayesian posterior analysis.

Future research work will be devoted towards enhancing different constituent parts of SLAYER. We foresee to resort to methods capable of learning topological relationships between multi-dimensional samples (e.g. Growing Neural Gas) for a better characterization of the patterns associated to each of the classes are not corpuscular (as in the `GEARS` dataset). Also, we will assess the extent to which we can extrapolate core SLAYER concepts (especially the prediction of the evolution of the concepts using Kalman) to other data stream learning paradigms, including online clustering. Likewise, improvements are planned in the way SLAYER connect the different clusters over mini-batches, for which an alternative measure of similarity between clusters more reliable than the distance between their centroids will be sought. Finally, we will explore whether mini-batches of variable size can be considered in the design of SLAYER, so that mini-batches are enlarged whenever the drift dynamics between the previous consecutive mini-batches are slow. This would lessen the computational effort required to run SLAYER, and could give rise to 1) a more accurate representation of the prevailing cluster distribution; and 2) a better traceability of the label-cluster assignment over time.

Acknowledgments

This work has received funding support from the Basque Government through the ELKARTEK program (3KIA project, ref. KK-2020/00049). J. Del Ser also acknowledges support from the Department of Education of the same institution (Consolidated Research Group MATHMODE, IT1294-19).

References

1. Arostegi, M., Torre-Bastida, A.I., Lobo, J.L., Bilbao, M.N., Del Ser, J.: Concept tracking and adaptation for drifting data streams under extreme verification la-

tency. In: International Symposium on Intelligent and Distributed Computing. pp. 11–25. Springer (2018)
2. Benavoli, A., Corani, G., Demšar, J., Zaffalon, M.: Time for a change: a tutorial for comparing multiple classifiers through bayesian analysis. The Journal of Machine Learning Research **18**(1), 2653–2688 (2017)
3. Dawid, A.P.: Present position and potential developments: Some personal views statistical theory the prequential approach. Journal of the Royal Statistical Society: Series A (General) **147**(2), 278–290 (1984)
4. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. The Journal of Machine Learning Research **7**, 1–30 (2006)
5. Dyer, K.B., Capo, R., Polikar, R.: Compose: A semisupervised learning framework for initially labeled nonstationary streaming data. IEEE transactions on neural networks and learning systems **25**(1), 12–26 (2013)
6. Elwell, R., Polikar, R.: Incremental learning of concept drift in nonstationary environments. IEEE Transactions on Neural Networks **22**(10), 1517–1531 (2011)
7. Ferreira, R.S., Zimbrão, G., Alvim, L.G.: Amanda: Semi-supervised density-based adaptive model for non-stationary data with extreme verification latency. Information Sciences **488**, 219–237 (2019)
8. Frederickson, C., Gracie, T., Portley, S., Moore, M., Cahall, D., Polikar, R.: Adding adaptive intelligence to sensor systems with mass. In: 2017 IEEE Sensors Applications Symposium (SAS). pp. 1–6. IEEE (2017)
9. Gama, J., Sebastião, R., Rodrigues, P.P.: Issues in evaluation of stream learning algorithms. In: Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining. pp. 329–338 (2009)
10. Gama, J., Sebastião, R., Rodrigues, P.P.: On evaluating stream learning algorithms. Machine learning **90**(3), 317–346 (2013)
11. Gama, J., Žliobaite, I., Bifet, A., Pechenizkiy, M., Bouchachia, A.: A survey on concept drift adaptation. ACM Computing Surveys **46**(4), 44 (2014)
12. Gomes, H.M., Read, J., Bifet, A., Barddal, J.P., Gama, J.: Machine learning for streaming data: state of the art, challenges, and opportunities. ACM SIGKDD Explorations Newsletter **21**(2), 6–22 (2019)
13. Kambatla, K., Kollias, G., Kumar, V., Grama, A.: Trends in big data analytics. Journal of Parallel and Distributed Computing **74**(7), 2561–2573 (2014)
14. Krempl, G.: The algorithm apt to classify in concurrence of latency and drift. In: International Symposium on Intelligent Data Analysis. pp. 222–233. Springer (2011)
15. Krempl, G., Žliobaite, I., Brzeziński, D., Hüllermeier, E., Last, M., Lemaire, V., Noack, T., Shaker, A., Sievi, S., Spiliopoulou, M., et al.: Open challenges for data stream mining research. ACM SIGKDD explorations newsletter **16**(1), 1–10 (2014)
16. Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE Transactions on Knowledge and Data Engineering **31**(12), 2346–2363 (2018)
17. Marrs, G.R., Hickey, R.J., Black, M.M.: The impact of latency on online classification learning with concept drift. In: International Conference on Knowledge Science, Engineering and Management. pp. 459–469. Springer (2010)
18. Razavi-Far, R., Hallaji, E., Saif, M., Ditzler, G.: A novelty detector and extreme verification latency model for nonstationary environments. IEEE Transactions on Industrial Electronics **66**(1), 561–570 (2019)
19. Settles, B.: Active learning. Synthesis lectures on artificial intelligence and machine learning **6**(1), 1–114 (2012)

20. Souza, V.M.A., Silva, D.F., Gama, J., Batista, G.E.A.P.A.: Data stream classification guided by clustering on nonstationary environments and extreme verification latency. In: SIAM International Conference on Data Mining (SDM). pp. 873–881. SIAM (2015)
21. Spiliopoulou, M., Ntoutsi, E., Theodoridis, Y., Schult, R.: Monic and followups on modeling and monitoring cluster transitions. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 622–626. Springer (2013)
22. Tsai, C.W., Lai, C.F., Chao, H.C., Vasilakos, A.V.: Big data analytics: a survey. Journal of Big Data **2**(1), 21 (2015)
23. Umer, M., Frederickson, C., Polikar, R.: Learning under extreme verification latency quickly: Fast compose. In: 2016 IEEE Symposium Series on Computational Intelligence (SSCI). pp. 1–8. IEEE (2016)
24. Umer, M., Polikar, R., Frederickson, C.: Level iw: Learning extreme verification latency with importance weighting. In: 2017 International Joint Conference on Neural Networks (IJCNN). pp. 1740–1747. IEEE (2017)
25. Yang, S.: Iot stream processing and analytics in the fog. IEEE Communications Magazine **55**(8), 21–27 (2017)

A Concept for Highly Automated Pre-Labeling via Cross-Domain Label Transfer for Perception in Autonomous Driving

Maarten Bieshaar[0000−0002−6471−6062], Marek Herde[0000−0003−4908−122X], Denis Huseljic[0000−0001−6207−1494], and Bernhard Sick[0000−0001−9467−656X]

University of Kassel, Wilhelmshöher Allee, 34121 Kassel, Germany
{mbieshaar,marek.herde,dhuseljic,bsick}@uni-kassel.de

Abstract. This article proposes a novel concept to leverage the time-consuming labeling process for training object detectors in automated driving. The approach uses pre-trained probabilistic, well-calibrated object detectors for different sensor modalities. Based on the knowledge about the sensor extrinsics, the probabilistic detections are transformed from one sensor modality into another. These transformed detections are then used as pre-labels for the respective sensor modality. However, these pre-labels are error-prone, such that we propose an additional dedicated labeling quality assessment. The latter allows us to attach a quality seal to automatically pre-labeled data sets and is the starting point for interactive human-in-the-loop learning.

Keywords: Highly Automated Pre-Labeling · Object Detection · Human-In-The-Loop Learning · Autonomous Driving · Imperfect Labels

1 Introduction

Artificial intelligence and, in particular, *machine learning* (ML) are the enabling technologies in autonomous driving. In this context, ML and deep learning techniques are already successfully used for perception, i.e., sensory environment and object recognition [4]. Training and validating these mostly deep neural networks, e.g., *convolutional neural networks* (CNN), requires vast amounts of labeled data. However, labeling, especially for object detection, is a time-consuming and, therefore, costly task [17]. In this article, we present an approach to significantly reduce the labeling effort in the particular application domain of ML-based object detection for highly automated driving. Our approach considers that many modern vehicles are equipped with various sensors, including cameras, LiDAR, and RADAR. We exploit this sensor diversity (i.e., the strengths and weaknesses of the respective sensors [12]) in our approach by transferring labels between different sensor modalities. First, we train object detectors for the single sensors. We further use these predictions as so-called pre-labels (i.e., imperfect, potentially error-prone labels). These can, in turn, be used to improve the object detectors of the other sensor modality. Our approach can be understood as semi-supervised cross-domain learning [2], whereas the object detectors are interpreted as multiple error-prone annotators [9]. However, in safety-critical applications such as highly automated driving, the created labelings must be quality-checked to ensure that no incorrect concepts are learned.

Contributions: We address this problem by proposing a detailed concept to automatically generate pre-labelings via cross-domain label transfer for perception in autonomous driving. Therefore, we identify four major research questions arising within our concept's stages and provide ideas for targeting each of them. We envision our concept as an application-driven starting point for human-in-the-loop learning. In this context, our concept provides methods leveraging interactive learning techniques in object detection, e.g., probabilistic object detectors, improving the labeling quality, coping with imperfect labels, and decreasing annotation effort. As another major contribution, we see the quality assessment of pre-labelings to support subsequent human-in-the-loop learning processes. In this sense, we aim to provide a quality seal to pre-labeled data sets, e.g., pre-labels having an expected mean average precision of 90%.

2 Highly Automated Pre-Labeling

This section describes the four stages of our concept (cf. Fig. 1) and formulates research questions. In the first stage, we develop probabilistic object detectors. The second stage aims to improve the calibration of these detectors further. Subsequently, the probabilistic predictions are interpreted as pre-labels of the different sensor modalities and optionally fused in the third stage. Finally, the concept is concluded by the fourth stage, including a labeling quality assessment based on probabilistic outputs. It serves as a starting point for human-in-the-loop learning to refine the pre-labels and release them for further model training.

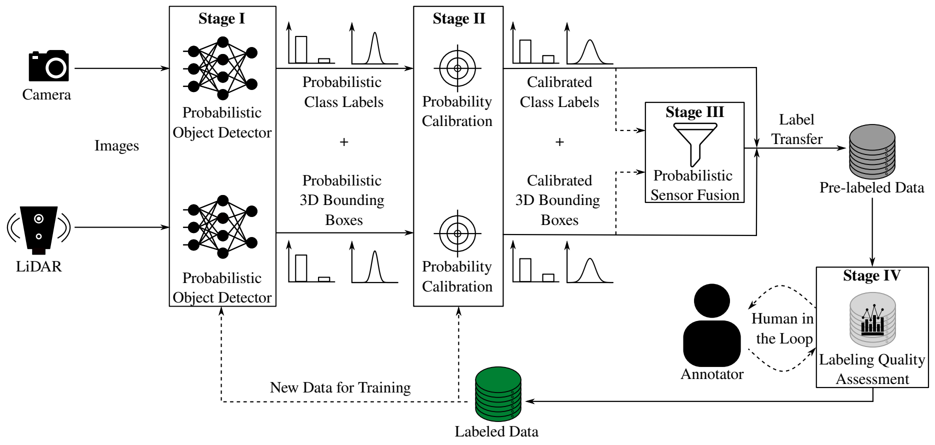


Fig. 1. Illustration of the proposed highly automated labeling process exemplary shown for two sensors. Dashed arrows represent optional processes.

Stage I: Probabilistic Object Detection – *How to train probabilistic object detectors for different sensor modalities?* In stage I, we consider pre-trained object detectors for different sensor domains, i.e., camera- and LiDAR-based object detectors [20, 21]. Common object detectors provide point estimates for the classification probabilities of the object and the coordinates of its 3D bounding box, i.e., position in space, the yaw angle, and its size [13]. In contrast, probabilistic object detectors provide predictive distributions for all quantities. Starting from pre-trained CNNs for 3D bounding box detection, e.g., for camera [21] and LiDAR [20], the main challenge will be a meaningful separation between aleatoric and epistemic uncertainty [10] without massively increasing computational complexity during training and inference. Therefore, we aim to leverage the approach proposed in [14], which enforces specific properties, i.e., smoothness and sensitivity in the feature space learned by a deep neural network. In this way, we can capture epistemic uncertainty by distributing that features and aleatoric uncertainty by evaluating the entropy of its predictive distribution.

Stage II: Probability Calibration – *How to further improve the calibration of the probabilistic detectors?* In stage II, we aim at improving probabilistic outputs by the object detectors as a foundation for the estimation of the labeling quality of the pre-labeling. For example, if the detector outputs a probability for a car with 90%, this statement should also be true in exactly 90% of the cases. The same applies to the probabilistic estimation of continuous target values, e.g., the coordinates of a 3D bounding box. However, deep neural networks tend to frequently output overconfident predictions, which can be alleviated through probability calibration methods [3, 15]. We intend to investigate post-hoc calibration methods, such as temperature scaling [5], and proper scoring rules to optimize the probabilistic object detectors [8].

Stage III: Label-Transfer and Probabilistic Sensor Fusion – *How to transfer labels between different sensor modalities and combine probabilistic predictions originating from different sensor modalities?* In stage III, we use the extrinsic sensor parameters to transfer pre-labels from one sensor domain to another. Therefore, we assume that the sensor extrinsics are known in advance. The transferred pre-labels can be used as labels for the other sensor modality and vice-versa. Moreover, we also investigate the fusion of probabilistic pre-labels (i.e., detections) originating from different sensor modalities (cf. stage I and II). The fusion is realized employing a Bayesian approach (cf. [6]). Therefore, we aim at examining using a joint probabilistic data-association filter [1] for the assignment of 3D bounding box detection from each sensor modality. Furthermore, we investigate the usage of Kalman and particle filters for object tracking [19].

Stage IV: Labeling Quality Assessment – *How to assess the label quality of probabilistic 3D bounding box predictions?* In stage IV, the aim is to assess the labeling quality [7, 16] of the obtained pre-labeled data set. For this purpose, we use the probabilistic predictions and determine expected values, e.g., with respect to the number of expected false classifications, the undetected objects, or the bounding box error. At this point, we want to explore the extent to which these expectations hold. Based on this, we aim to derive a quality seal for the pre-labeled data set. In this context, a starting point for the pre-labeling quality estimation is the ML-based online performance estimation using multiple sensors [11]. Moreover, we expect to assess whether a camera can be used for labeling LiDAR and vice-versa and when such a label transfer is useful. Ideally, the final labeling quality assessment supports human experts to decide whether individual bounding boxes need to be re-labeled (cf. active learning [18]) or whether the pre-labeling is of sufficient quality to release the pre-labeled dataset for further processing such as model training.

3 Conclusion

This article presents a novel concept for highly automated pre-labeling via cross-domain label transfer for perception in autonomous driving. The novelty of our concept lies in the label transfer exploiting the strengths and weaknesses of different sensor modalities for object detection. The use of multiple sensors to improve perception is not new. However, the use for pre-labeling (in the context of 3D bounding box detection) in combination with an explicit quality assessment component under consideration of calibrated probabilistic predictions represents a novel approach. It allows us to attach a quality seal to pre-labeled data sets. The quality assessment is the starting point for human-in-the-loop learning and iterative model improvement. Although our concept focuses on the autonomous driving domain with LiDAR and camera sensors, it can be extended toward multiple sensors and possibly different applications involving data from multiple sensors. Moreover, the presented ideas form a foundation for further investigations in the area of interactive adaptive learning. For example, the uncertainty estimates of the developed probabilistic object detectors might be used to derive novel utility measures for active learning in object detection.

Acknowledgments

This work results from the project KI Data Tooling (19A20001O) funded by German Federal Ministry for Economic Affairs and Energy (BMWI) and the DeCoInt^2-project financed by the German Research Foundation (DFG) within the priority program SPP 1835: “Kooperativ interagierende Automobile”, grant number SI 674/11-2. We, in particular, thank Kristina Dingel, Alice Moallemy-Oureh, and Daniel Kottke for their insightful feedback.

References

1. Bar-Shalom, Y., Daum, F., Huang, J.: The probabilistic data association filter. IEEE Control Systems Magazine **29**(6), 82–100 (2009)
2. van Engelen, J.E., Hoos, H.: A survey on semi-supervised learning. Machine Learning **109**, 373–440 (2019)
3. Feng, D., Rosenbaum, L., Gläser, C., Timm, F., Dietmayer, K.: Can We Trust You? On Calibration of a Probabilistic Object Detector for Autonomous Driving. In: IEEE/RSJ IROS Workshops. Macau, China (2019)
4. Feng, D., Haase-Schütz, C., Rosenbaum, L., Hertlein, H., Gläser, C., Timm, F., Wiesbeck, W., Dietmayer, K.: Deep multi-modal object detection and semantic segmentation for autonomous driving: Datasets, methods, and challenges. IEEE T-ITS **22**(3), 1341–1360 (2021)
5. Guo, C., Pleiss, G., Sun, Y., Weinberger, K.Q.: On Calibration of Modern Neural Networks. In: ICML. pp. 1321–1330. Sydney, NSW, Australia (2017)
6. Gustafsson, F.: Statistical Sensor Fusion. Stud.Lit. AB, Lund, Sweden (2015)
7. Haase-Schütz, C., Hertlein, H., Wiesbeck, W.: Estimating labeling quality with deep object detectors. In: IV. pp. 33–38. Paris, France (2019)
8. Harakeh, A., Waslander, S.L.: Estimating and Evaluating Regression Predictive Uncertainty in Deep Object Detectors. In: ICLR (2021)
9. Herde, M., Kottke, D., Huseljic, D., Sick, B.: Multi-Annotator Probabilistic Active Learning. In: ICPR. pp. 10281–10288. Milan, Italy (2021)
10. Huseljic, D., Sick, B., Herde, M., Kottke, D.: Separation of aleatoric and epistemic uncertainty in deterministic deep neural networks. In: ICPR. pp. 9172–9179. Milan, Italy (2021)
11. Klingner, M., Bär, A., Mross, M., Fingscheidt, T.: Improving Online Performance Prediction for Semantic Segmentation. In: CVPR Workshop Safe Artificial Intelligence for Automated Driving. p. 8 (2021)
12. Kowol, K., Rottmann, M., Bracke, S., Gottschalk, H.: YOdar: Uncertainty-based Sensor Fusion for Vehicle Detection with Camera and Radar Sensors. In: ICAART. pp. 177–186 (2021)
13. Mousavian, A., Anguelov, D., Flynn, J., Košecká, J.: 3D Bounding Box Estimation Using Deep Learning and Geometry. In: CVPR. pp. 5632–5640 (2017)
14. Mukhoti, J., Kirsch, A., van Amersfoort, J., Torr, P.H., Gal, Y.: Deterministic neural networks with appropriate inductive biases capture epistemic and aleatoric uncertainty. arXiv preprint arXiv:2102.11582 (2021)
15. Niculescu-Mizil, A., Caruana, R.: Predicting good probabilities with supervised learning. In: ICML. pp. 625–632. New York, NY (2005)
16. Northcutt, C., Jiang, L., Chuang, I.: Confident Learning: Estimating Uncertainty in Dataset Labels. Journal of Artificial Intelligence Research **70**, 1373–1411 (2021)
17. Papadopoulos, D.P., Uijlings, J.R.R., Keller, F., Ferrari, V.: Extreme Clicking for Efficient Object Annotation. In: ICCV. pp. 4940–4949 (2017)
18. Settles, B.: Active Learning Literature Survey. Tech. rep., University of Wisconsin-Madison (2010)
19. Thrun, S., Burgard, W., Fox, D. (eds.): Probabilistic Robotics. Intelligent Robotics and Autonomous Agents, MIT Press, Cambridge, MA (2005)
20. Yin, T., Zhou, X., Krähenbühl, P.: Center-based 3D Object Detection and Tracking. CVPR (2021)
21. Zhou, X., Wang, D., Krähenbühl, P.: Objects as Points. arXiv preprint arXiv:1904.07850 (2019)

Active Class Selection with Uncertain Deployment Class Proportions

Mirko Bunse(✉) and Katharina Morik

TU Dortmund University, Artificial Intelligence Group, 44221 Dortmund, Germany
{firstname.lastname}@tu-dortmund.de

Abstract. Active class selection strategies actively choose the class proportions of the data with which a classifier is trained. While this freedom of choice can improve the classification accuracy and reduce the data acquisition cost, it has also motivated theoretical studies that quantify the limited trustworthiness of the resulting classifier when the chosen class proportions differ from the class proportions that need to be handled during deployment. In this work, we build on these theoretic foundations to propose an active class selection strategy that allows machine learning practitioners to express their prior beliefs about the deployment class proportions. Unlike existing approaches, our strategy is justified by PAC learning bounds and naturally supports any degree of uncertainty with respect to these prior beliefs.

Keywords: Active class selection · Imbalanced binary classification · PAC learning theory.

1 Introduction

Active class selection (ACS) [11, 9] allows machine learning practitioners to actively choose the label proportions of their training data. This freedom of choice is due to a *class-conditional* data generator, e.g. an experiment or a simulation, which acquires feature vectors for arbitrarily chosen classes. Data generators of this kind appear in various use cases, such as astro-particle physics [4, 3], gas sensor arrays [11], and brain computer interaction [13].

Lomasky et al. [11] have put forward the idea that such a generator can be leveraged in a sequence of multiple acquisition steps, as sketched in Fig. 1. In each step, a classifier is trained and evaluated on all examples that have been acquired so far, starting from a small initial data set (i). Based on the classifier's performance, a data acquisition *strategy* is then allowed to choose the label proportions of the next acquisition step (ii). The class-conditional data generator realizes these proportions, i.e. it produces a batch of labeled data according to the choice of the strategy (iii). This batch adds to the training set from which the classifier will be trained in all subsequent iterations. The promise of such a sequential and informed data acquisition is that the classifier can benefit in terms of data acquisition cost and performance, as compared to being trained with some predetermined proportions of classes.

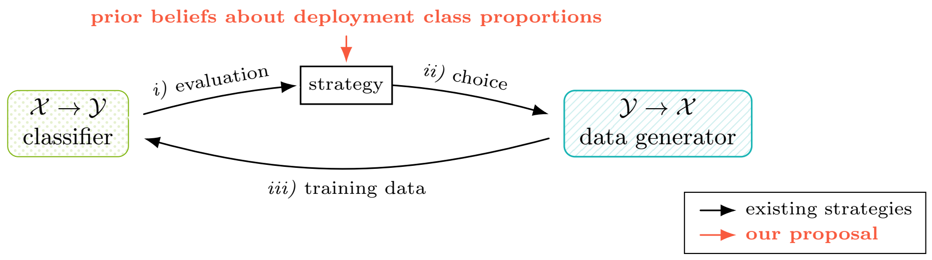

prior beliefs about deployment class proportions
i) evaluation
strategy
ii) choice
$\mathcal{X} \to \mathcal{Y}$
classifier
$\mathcal{Y} \to \mathcal{X}$
data generator
iii) training data
existing strategies
our proposal

Fig. 1. Strategies for active class selection choose the label proportions of newly acquired training data. They are allowed to base their decisions on the performance of a classifier that is trained with all previously acquired data. We propose to incorporate prior beliefs, which can be uncertain, into the decision making.

Existing strategies [9, 11] for ACS do not account for the class proportions that a trained model needs to handle during deployment; they solely focus on the perceived *difficulty* of classes. One notable exception is a strategy that acquires training data precisely with those label proportions that are faced in the deployment stage; by design, this strategy requires the practitioner to know the deployment class proportions precisely in advance. However, what if we know the deployment class proportions not precisely, but with some degree of uncertainty? For instance, astro-particle physicists can estimate the ratio between their signal and their background class only roughly, as being approximately $1 : 10^3$ or even $1 : 10^4$ [2]. We are not aware of any ACS strategy that supports uncertain deployment class proportions out of the box.

Motivated by such uncertainties, we have recently proposed a theoretically justified *certificate* for ACS-trained models [4]. This certificate declares a range of deployment class proportions for which a given model is accurate (i.e. has an ACS-induced error smaller than some $\varepsilon > 0$) with a high probability (i.e. with probability at least $1 - \delta$). This declaration can help practitioners in assessing the practical value of an ACS-trained model. However, it has no immediate implication on how to acquire data—in terms of an ACS strategy—when the deployment class proportions are uncertain.

In the following, we therefore evolve the theoretical basis of our certificate towards a data acquisition strategy for ACS. This strategy uniquely combines the following qualities:

- our ACS strategy naturally supports uncertainty about the deployment class proportions, e.g. as expressed by a Beta prior for binary classification.
- our strategy is theoretically justified by PAC learning bounds.

Our experiments suggest that our strategy, even under high amounts of uncertainty, exhibits a performance that is comparable to the performance of an optimal strategy with privileged access to the class proportions of the test set. Other strategies, which are oblivious to the deployment class proportions, fall behind by a significant margin.

We summarize the theoretic foundations of ACS in Sec. 2 before we detail our strategy in Sec. 3. The experiments in Sec. 4 lead to our conclusion in Sec. 5.

2 Theoretical Background

The term "domain", as proposed by domain adaptation [14, 12], describes a probability density function over the data space $\mathcal{X} \times \mathcal{Y}$. In ACS, we assume that the *source domain* $\mathcal{S}$—where a machine learning model is trained—differs from the *target domain* $\mathcal{T}$—where the model is deployed—only in terms of the class proportions $p_{\mathcal{S}} \neq p_{\mathcal{T}}$. Such deviations occur due to the freedom of ACS strategies to choose any $p_{\mathcal{S}}$ for the acquisition of training data. We are interested in the impact of such deviations on the deployment performance, i.e. on the classification performance with respect to $\mathcal{T}$.

Recently, a PAC learning perspective on this setting has provided us with Theorem 1 [4]. This result quantifies the difference in loss values $L(h)$ between an ACS-generated training set D and the target domain $\mathcal{T}$. Only if this difference is small, we can expect to learn a classifier h from D that is accurate also with respect to $\mathcal{T}$, similar to standard PAC learning theory. The key insight of this theorem is that the relevant loss difference between D and $\mathcal{T}$ is continuously approaching the inter-domain gap $\Delta p \cdot \Delta \ell$ while the training set size m increases. In ACS, this increase happens naturally while more and more data is actively being acquired, so that the error of any ACS-trained classifier is increasingly dominated by this gap. Here, $\Delta p = |p_{\mathcal{T}} - p_{\mathcal{S}}|$ denotes the difference between class proportions and $\Delta \ell = |\ell_{Y=2}(h) - \ell_{Y=1}(h)|$ denotes the difference between class-wise losses. The latter of these terms is constant across domains $\mathcal{S}$ and $\mathcal{T}$. In turn, $\Delta p \cdot \Delta \ell$ is constant with respect to the random draw of the training set D and is therefore independent of ε, δ, and m; it reflects the interplay between the classifier h, the data distribution, and the loss function.

Theorem 1 (Identical mechanism bound; binary classification [4]). *For any $\varepsilon > 0$, any $h \in \mathcal{H}$, with probability at least $1 - \delta$, where $\delta = 4e^{-2m\varepsilon^2}$:*

$$\Delta p \cdot \Delta \ell - \varepsilon \quad \leq \quad |L_{\mathcal{T}}(h) - L_D(h)| \quad \leq \quad \Delta p \cdot \Delta \ell + \varepsilon$$

The true difference $\Delta \ell$ from Theorem 1 is unknown, but we can estimate an upper bound $\Delta \ell^*$ of this quantity from ACS-generated data. The details on this estimation are already presented in the scope of ACS model certification [4] and do not need to be repeated here. All we need to know to establish our ACS strategy is that $\Delta \ell^*$ is the smallest upper bound of $\Delta \ell$ that holds with probability at least $1 - \delta$. The probabilistic nature of this upper bound stems from the fact that $\Delta \ell^*$ is estimated from finite amounts of data.

3 A Strategy for Uncertain Class Proportions

The goal of our strategy is to decrease the inter-domain gap $\Delta p \cdot \Delta \ell$ from Theorem 1 as much as possible, as according to a prior distribution $\hat{\mathbb{P}}$ of the deployment class proportions $p_{\mathcal{T}}$. This goal will allow any binary classification algorithm to learn accurate predictions for the target domain, as according to the prior beliefs of a domain expert.

Formally, we assume a prior $\hat{\mathbb{P}} : [0,1] \to [0,1]$ of the positive class prevalence $p_{\mathcal{T}} \in [0,1]$ to be given. We incorporate $\hat{\mathbb{P}}$ by marginalizing the inter-domain gap over this prior, as according to Eq. 1. Since we do not know the true $\Delta\ell$, we are using the estimated upper bound $\Delta\ell^*$ instead. Consequently, the marginalization according to $\Delta\ell^*$ is an upper bound, with probability $1-\delta$, of the marginalization according to the true $\Delta\ell$.

$$\varepsilon^* \;=\; \int_0^1 \hat{\mathbb{P}}(p_{\mathcal{T}} = p) \,\cdot\, \underbrace{|p_{\mathcal{S}} - p|}_{=\,\Delta p} \cdot \Delta\ell^* \;\; \mathrm{d}\,p \tag{1}$$

In each ACS iteration, we are free to alter the class proportions $p_{\mathcal{S}}$ of the ACS-generated training set to some degree, depending on how much data we acquire in each batch and on how much data we already have acquired. In fact, we can understand $p_{\mathcal{S}} = \frac{m_2}{(m_1+m_2)}$ as a function of the class-wise numbers of samples m_1 and m_2. The upper bound $\Delta\ell^*$ also lends itself for being interpreted as a function of sample sizes: the more data is acquired in both classes, the tighter will our estimation of this quantity be. Ultimately, we consider ε^* to be a function of m_1 and m_2, so that we can minimize ε^* via an optimal choice of m_1 and m_2 in each data acquisition batch.

3.1 Minimizing the Marginalized Error

Our strategy decreases ε^* in the direction of its steepest descent, i.e. it takes a simple gradient step with respect to the acquisition vector $\boldsymbol{m} = (m_1, m_2)$. The gradient which defines the steepest descent is computed via the product rule:

$$\begin{aligned} \nabla_{\boldsymbol{m}}\, \varepsilon^* \;&=\; \nabla_{\boldsymbol{m}} f \cdot \Delta\ell^* \;+\; f \cdot \nabla_{\boldsymbol{m}} \Delta\ell^* \\ \text{where } f(\boldsymbol{m}) \;&=\; \int_0^1 \hat{\mathbb{P}}(p_{\mathcal{T}} = p) \,\cdot\, |p_{\mathcal{S}}(\boldsymbol{m}) - p| \;\; \mathrm{d}\,p \end{aligned} \tag{2}$$

We will come back to the function f shortly. For now, we plug $\Delta\ell^*$ and $\nabla_{\boldsymbol{m}}\Delta\ell^*$ into the equation above. These functions are defined by

$$\begin{aligned} \Delta\ell^*(\boldsymbol{m}) \;&=\; \hat{\ell}_{Y=2}(h) + \sqrt{\frac{\ln \delta_2}{-2m_2}} - \hat{\ell}_{Y=1}(h) + \sqrt{\frac{\ln \delta_1}{-2m_1}}, \\ [\nabla_{\boldsymbol{m}} \Delta\ell^*]_y \;&=\; \left(-\frac{\ln \delta_y}{m_y}\right)^{\frac{3}{2}} \cdot (2\sqrt{2} \ln \delta_y)^{-1}, \end{aligned} \tag{3}$$

where the δ_y are probabilities of violations of $\Delta\ell^*$ that occur from either one of the class-wise losses $\ell_{Y=y}(h)$ in $\Delta\ell$. In fact, finding a suitable assignment of δ_y values within a given probability budget $\delta = \delta_1 + \delta_2 - \delta_1\delta_2$ is the central difficulty in model certification; there, the sample size $\boldsymbol{m}$ is fixed, so that $\Delta\ell^*$ can be optimized over this assignment [4]. Here, we keep the δ_y fixed instead, to values that are obtained with a certificate from previous ACS acquisitions. This change allows us to optimize $\Delta\ell^*$ over $\boldsymbol{m}$ to acquire new data and it guarantees that $\Delta\ell^*$ remains an upper bound of the true $\Delta\ell$ also in the next batch, at least with probability $1-\delta$. The class-wise estimates $\hat{\ell}_{Y=y}(h)$ in Eq. 3 are the average values of losses in the training data; they are also part of our certificate.

3.2 A Beta Prior for Binary Class Proportions

Now we turn to the function value and the gradient of the function f in Eq. 2. Plugging a parametric prior $\hat{\mathbb{P}}$ into this function can allow us to compute these terms efficiently, in closed forms. To this end, a Beta(α, β) prior is suitable for binary classification because the Beta distribution is a conjugate prior of the Bernoulli distribution, which in turn is a suitable model for the prevalence of binary class labels. As a matter of convenience, the parameters $\alpha > 0$ and $\beta > 0$ can be chosen such that the resulting distribution has some predetermined mean and standard deviation; we believe that domain experts can often express their prior beliefs in terms of these properties.

Plugging a Beta prior into the f function from Eq. 2 yields the following components, where I is the regularized incomplete Beta function:

$$\begin{aligned} f_{\alpha,\beta}(\boldsymbol{m}) &= \frac{2p_{\mathcal{S}}(\boldsymbol{m})^{\alpha}(1-p_{\mathcal{S}}(\boldsymbol{m}))^{\beta}}{(\alpha+\beta)B(\alpha,\beta)} + \left(p_{\mathcal{S}}(\boldsymbol{m}) - \frac{\alpha}{\alpha+\beta}\right)\left(2I_{p_{\mathcal{S}}(\boldsymbol{m})}(\alpha,\beta)-1\right) \\ \nabla_{\boldsymbol{m}} f_{\alpha,\beta} &= \frac{2I_{p_{\mathcal{S}}(\boldsymbol{m})}(\alpha,\beta)-1}{(m_1+m_2)^2} \cdot \begin{pmatrix} m_2 \\ -m_1 \end{pmatrix} \end{aligned} \tag{4}$$

Plugging Eq. 3 and 4 into Eq. 2 provides us with a gradient that we can compute analytically from a certificate with a δ_y assignment, from sample sizes m_1 and m_2 and from the prior parameters α and β. The negative gradient $-\nabla_{\boldsymbol{m}}\,\varepsilon^*$ of the marginalized error ε^* defines the class-wise numbers of samples that our strategy acquires in the next data acquisition batch.

With small data volumes or with highly imbalanced classes, our strategy is dominated by the $\Delta\ell^*$ component; small classes need additional data until this upper bound holds with some desired probability $1-\delta$. Constrastingly, when the total data volume is large, our strategy is dominated by the f component; to this end, a Beta prior favors class proportions that are close to its mean $\frac{\alpha}{\alpha+\beta}$. The turning point between these two behaviors is well-founded in the PAC learning theory that underlies the estimation of $\Delta\ell^*$.

4 Experiments

The first introduction of the ACS problem is already accompanied by the proposal of five heuristic ACS strategies [11]. In the following, we compare our own strategy from Sec. 3 to these five heuristics:

proportional: always sample according to $p_{\mathcal{T}}$, provided that these true proportions are already known at training time.

uniform: always sample all classes in the same amount.

inverse: sample according to the inverse accuracy of a classifier that is trained on earlier batches; the underlying assumption is that weak class-wise performances can be counteracted with over-sampling.

improvement: sample according to the class-wise improvement in accuracy that has occurred between the current iteration and the iteration before; this strategy assumes that stable performances, i.e. performances that did not change recently, will remain stable during future acquisitions.

redistriction: sample according to the class-wise number of training examples for which the prediction has changed between the current iteration and the iteration before; the assumption here is that stability can be promoted by over-sampling classes with volatile decision boundaries.

Our theoretical analysis of the ACS problem [5, 4] reveals that the *proportional* strategy is actually more than a heuristic; this strategy is indeed *optimal* in the limit of data acquisition. However, it requires precise knowledge of $p_\mathcal{T}$, which practitioners might not be able to provide. Contrastingly, all other strategies are entirely oblivious to the deployment proportions; they solely focus on different notions of class-wise difficulties.

This shortcoming is also shared by an ACS strategy that aggregates utility scores of pseudo-instances [9]. For now, we have excluded this approach from our comparison, due to this property. For future work, however, we expect that the method can overcome this limitation with a recent update of its utility function [8]. This update supports a prior of $p_\mathcal{T}$, which is in line with our idea of incorporating prior beliefs in ACS. Embedding the update in the original pseudo-instance strategy, however, might not be trivial.

4.1 Methodology

We have parameterized the Beta prior of our strategy with a predetermined mean and standard deviation, both set to the true value of $p_\mathcal{T}$. Accordingly, the mean of the prior is well aligned with the true class proportions of the deployment data; the uncertainty, however, is as large as possible.

In accordance to a reliable evaluation methodology [7], we present pairwise differences between ACS strategies in terms of their statistical significance. A comprehensive way of plotting such differences is through critical difference diagrams [6, 1], which compare multiple strategies over multiple data sets in a statistically sound way. We employ accuracy as the underlying performance metric and we conduct multiple trials to obtain an average performance value for each combination of strategy and data set. These average performances are then summarized through critical difference diagrams.

We define the trials via five repetitions of a three-fold cross validation. From the *imbalanced-learn*[1] package [10], we retrieve 13 data sets that have at least 150 minority class samples (to facilitate sampling) and at most 100 features (to

[1] `https://imbalanced-learn.org/stable/datasets/`

facilitate learning). We ensure comparability between all strategies by employing the same classifier in all experiments, a logistic regression with default meta-parameters. The data acquisition happens in up to 8 batches, each of which acquires 50 new training examples. However, not all strategies reach the last batch on all data sets; we stop each trial as soon as the strategy exhausts one of the classes. We opted for this early stopping criterion to focus on “realistic” acquisitions that happen due to free choices and not due to the fact that our experiment only simulates class-dependent data acquisition with finite pools of data. For the same reason, and due to weak performances on imbalanced data, we did not evaluate the *uniform* strategy here. Due to the early stopping, it becomes increasingly harder to detect significant differences; while the batches three and four can be evaluated on all data sets, only 9 data sets remain for batch eight. The implementation of our configurable experiments is available online[2].

4.2 Results and Discussion

Fig. 2 presents the critical difference diagrams, as according to our evaluation methodology. We see that our method, with access to an uncertain prior of $p_{\mathcal{T}}$, performs as well as the privileged strategy that knows $p_{\mathcal{T}}$ precisely. Moreover, our method outperforms all existing strategies which are oblivious to $p_{\mathcal{T}}$.

Fig. 3 traces this success back to the acquisition behavior that each strategy exhibits. Our own strategy quickly approaches the true proportions $p_{\mathcal{T}}$ of classes, due to the perfect alignment between the mean of the prior and $p_{\mathcal{T}}$. For the particular case of a Beta prior, this behavior is a reason for concern: if the mean of this prior was not well aligned with $p_{\mathcal{T}}$, we might have acquired data in mistaken class proportions; only if the mean of the Beta prior is sufficiently accurate, we can expect the competitive behavior that Fig. 2 suggests. Future research down this lane, e.g. with other types of prior distributions, is needed.

Fig. 3 further reveals two explanations for the poor performances of the existing strategies: first, all of these strategies exhibit a central tendency of staying close to the class proportions of the initial training set; second, each of these strategies prefers class proportions of an increasingly large variability. Both of these behaviors are due to the sole focus of these strategies on the perceived difficulties of classes, which can differ considerably between the data sets.

5 Conclusion and Outlook

In contrast to existing ACS strategies, which either assume precise knowledge about the deployment class proportions or no knowledge at all, we have advocated the incorporation of a prior distribution that expresses beliefs about the class proportions with any degree of (un)certainty. Our ACS strategy is well-founded on PAC learning bounds which we have recently proposed for ACS [4]. Experiments suggest that our strategy performs as well as the fully certain case, which, however, is harder to specify than an uncertain prior.

[2] `https://github.com/mirkobunse/AcsCertificates.jl`

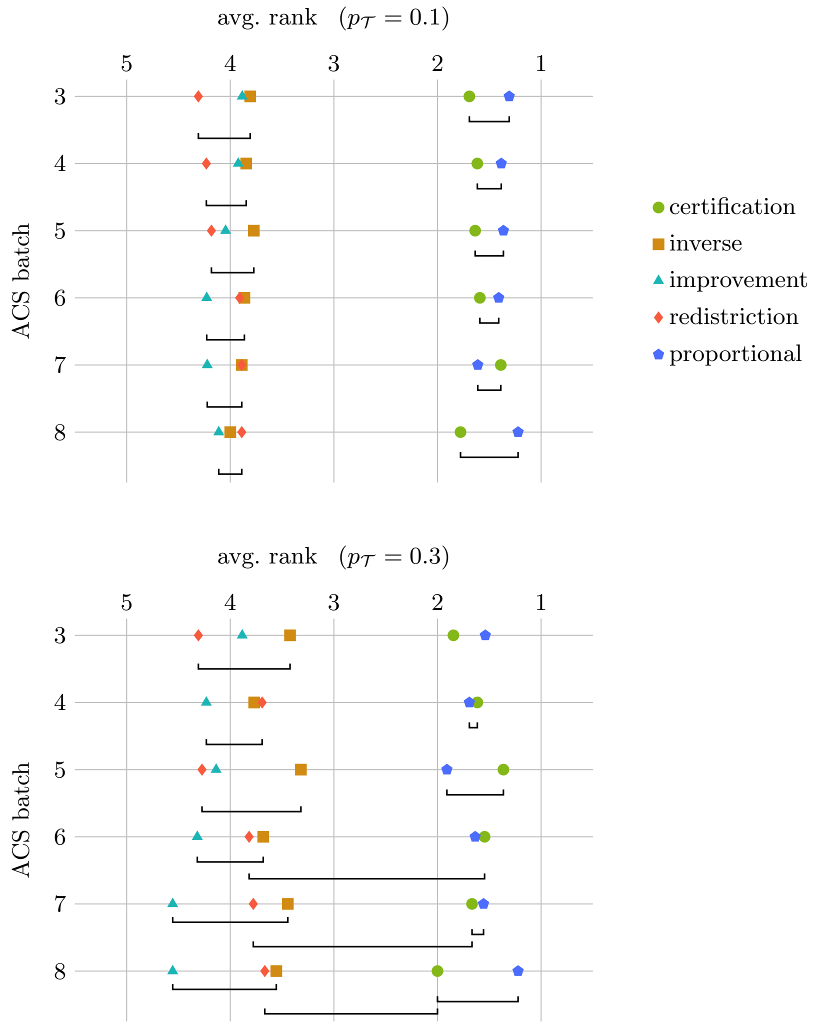


Fig. 2. Critical difference diagrams evaluate our ACS strategy (●) against existing ACS strategies [11], one of which has privileged access to the true class proportions $p_{\mathcal{T}}$ (⬟). The two plots present different values of $p_{\mathcal{T}}$. Each position on the vertical axes corresponds to one critical difference diagram for one batch in the ACS data acquisition loop. Horizontal positions correspond to the average ranks of strategies across multiple data sets, as according to the average accuracy in multiple trials; lower ranks are better. Horizontal connections between two or more strategies indicate that a Wilcoxon signed-rank test is not able to detect significant differences between these methods from the performances they exhibit.

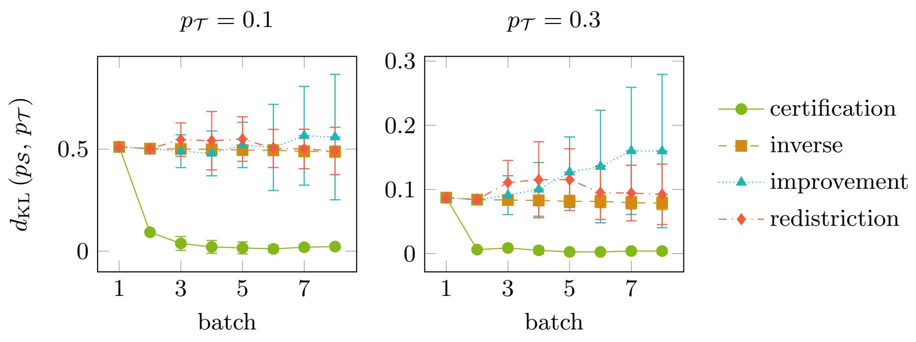


Fig. 3. Our ACS strategy (●) quickly approaches the true proportions $p_{\mathcal{T}}$ of classes in terms of the Kullback-Leibler divergence d_{KL}. Due to the uncertainty of the prior, however, this divergence always remains above zero. The standard deviation of d_{KL}, as displayed by the error bars, increases considerably with the other strategies.

Future work on ACS should focus on strategies that support multi-class classification and regression. We identify the PAL-ACS framework [9] with a recent update of its utility function [8] as a promising candidate in this direction.

Acknowledgments

This work has been supported by Deutsche Forschungsgemeinschaft (DFG) within the Collaborative Research Center SFB 876 "Providing Information by Resource-Constrained Data Analysis", project C3, and by the Federal Ministry of Education and Research of Germany as part of the competence center for machine learning ML2R (01IS18038 A/B).

References

1. Benavoli, A., Corani, G., Mangili, F.: Should we really use post-hoc tests based on mean-ranks? J. Mach. Learn. Res. **17**(1), 152–161 (2016)
2. Bockermann, C., Brügge, K., Buss, J., Egorov, A., Morik, K., Rhode, W., Ruhe, T.: Online analysis of high-volume data streams in astroparticle physics. In: Europ. Conf. on Mach. Learn. and Knowledge Discovery in Databases. Springer (2015)
3. Bunse, M., Bockermann, C., Buss, J., Morik, K., Rhode, W., Ruhe, T.: Smart control of Monte Carlo simulations for astroparticle physics. In: Astronomical Data Analysis Software and Systems. pp. 417–420 (2017)
4. Bunse, M., Morik, K.: Certification of model robustness in active class selection. In: Europ. Conf. on Mach. Learn. and Knowledge Discovery in Databases. Springer (2021)
5. Bunse, M., Weichert, D., Kister, A., Morik, K.: Optimal probabilistic classification in active class selection. In: Int. Conf. on Data Mining. IEEE (2020)
6. Demšar, J.: Statistical comparisons of classifiers over multiple data sets. J. Mach. Learn. Res. **7**(1), 1–30 (2006)

7. Kottke, D., Calma, A., Huseljic, D., Krempl, G., Sick, B.: Challenges of reliable, realistic and comparable active learning evaluation. In: Workshop and Tutorial on Interactive Adaptive Learn. pp. 2–14 (2017)
8. Kottke, D., Herde, M., Sandrock, C., Huseljic, D., Krempl, G., Sick, B.: Toward optimal probabilistic active learning using a Bayesian approach. Mach. Learn. **110**(6), 1199–1231 (2021)
9. Kottke, D., Krempl, G., Stecklina, M., von Rekowski, C.S., Sabsch, T., Minh, T.P., Deliano, M., et al.: Probabilistic active learning for active class selection. In: NeurIPS Workshop on the Future of Interactive Learn. Mach. (2016)
10. Lemaitre, G., Nogueira, F., Aridas, C.K.: Imbalanced-learn: A python toolbox to tackle the curse of imbalanced datasets in machine learning. J. Mach. Learn. Res. **18** (2017)
11. Lomasky, R., Brodley, C.E., Aernecke, M., Walt, D., Friedl, M.A.: Active class selection. In: Europ. Conf. on Mach. Learn. and Knowledge Discovery in Databases. Springer (2007)
12. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Trans. Knowl. Data Eng. **22**(10) (2010)
13. Parsons, T.D., Reinebold, J.L.: Adaptive virtual environments for neuropsychological assessment in serious games. IEEE Trans. Consumer Electron. **58**(2) (2012)
14. Wang, M., Deng, W.: Deep visual domain adaptation: A survey. Neurocomputing **312** (2018)

Sample Noise Impact on Active Learning

Alexandre Abraham$^{1[0000-0003-3693-0560]}$
and Léo Dreyfus-Schmidt$^{1[0000-0001-8271-1217]}$

Dataiku, Paris, France
{alexandre.abraham,leo.dreyfus-schmidt}@dataiku.com

Abstract. This work explores the effect of noisy sample selection in active learning strategies. We show on both synthetic problems and real-life use-cases that knowledge of the sample noise can significantly improve the performance of active learning strategies. Building on prior work, we propose a robust sampler, *Incremental Weighted K-Means* that brings significant improvement on the synthetic tasks but only a marginal uplift on real-life ones. We hope that the questions raised in this paper are of interest to the community and could open new paths for active learning research.

1 Introduction

When training machine learning models, data quality is undoubtedly the most fundamental requirement. A recent study [5] has shown that pervasive errors in the test set of famous datasets could lead to selecting a suboptimal model. In active learning, where a small number of samples are selected to be labeled by an oracle, it becomes paramount as selecting samples of poor quality may worsen the model's performance.

Sample diversity in the training set is also essential and has been the main focus of recent active learning strategies. Performance improvements come from new ways of combining uncertainty and diversity in a single framework. Batch-BALD [4] adds diversity by minimizing the joint mutual information between batch samples. Core-sets [6] and [8] use a clustering approach to scatter the selected samples across the sample space. The method proposed in [3] minimizes the similarity between the samples of the batch while minimizing the similarity with already labeled samples. The most common explanation for the observed performance uplift when enforcing diversity is that a homogeneous set of samples contains much redundant information while a diverse one informs the model with several classification patterns.

Enforcing diversity entails selecting samples where uncertainty is not maximal. Therefore, the selected samples are further away from the decision boundary and easier to classify. We hypothesize that this side-effect of diversity contributes to its success. In classification, mislabeled or very ambiguous samples – like five that looks like six in MNIST – can be detrimental to the model [5]. As the density of such samples is higher near the classification boundary, we increase the chances of obtaining meaningful samples by selecting samples further away.

This paper proposes a metric to evaluate the quantity of such noisy samples in a dataset, and we design a query strategy to avoid them. We first validate our approach by showing the existence of these samples on a synthetic example and observe that diversity-based methods are less likely to select those. We show that our results obtained on synthetic data do not generalize well to real tasks, propose an explanation and ideas to mitigate the problem.

2 Sample-noise robust strategies

In the following, $\mathcal{D}$ designates a dataset and h a probabilistic classifier. A subscript indicates the nature of datasets: L stands for labeled samples, U unlabeled, T test, and B designates a batch of samples. Iterations are indicated with a superscript when pertinent.

2.1 The pervasiveness of sample noise

In his seminal work on active learning, Settles [7] defines the most valuable samples at iteration i as the one with the lowest maximum predicted probability among classes:

$$\text{lowest\_confidence(x)} = 1 - h_1^i(x)$$

With h_k^i being the k-th probability predicted by the classifier learned at iteration i in descending order, so that h_1^i is the maximum predicted probability at iteration i. This definition assumes that each sample can reach a predicted probability of 1. The difference between 1 and the predicted probability represents the information that the model is expected to gain when the sample gets labeled.

However, classifiers do not always reach a predicted probability of 1 for all samples. Fig. 1 shows the distribution of predicted probabilities on various standard tasks (see details in section 3). If some datasets like LDPA present an almost uniform distribution, MNIST is very polarized towards 1 while having outliers below 0.5.

We call noisy the samples located at the boundary between two classes, which commonly have a low predicted probability for their class. Noisy samples can be due to signal noise in the data that makes them hard-to-classify, labeling errors, or to a genuine ambiguity such as a four that looks like a nine in MNIST (see Fig. 1, right). Noisy samples are a challenge in active learning as they may get overly selected by uncertainty-based methods despite their low quality. At a given iteration of an active learning experiment, noisy samples occur for two reasons. First, those samples may be easy to classify, but our current classifier lacks the knowledge to do so. Labeling this sample could be useful as it would help the model determine if the ideal decision boundary is close or not. This type of uncertainty is called *epistemic* and can be reduced with more samples. However,

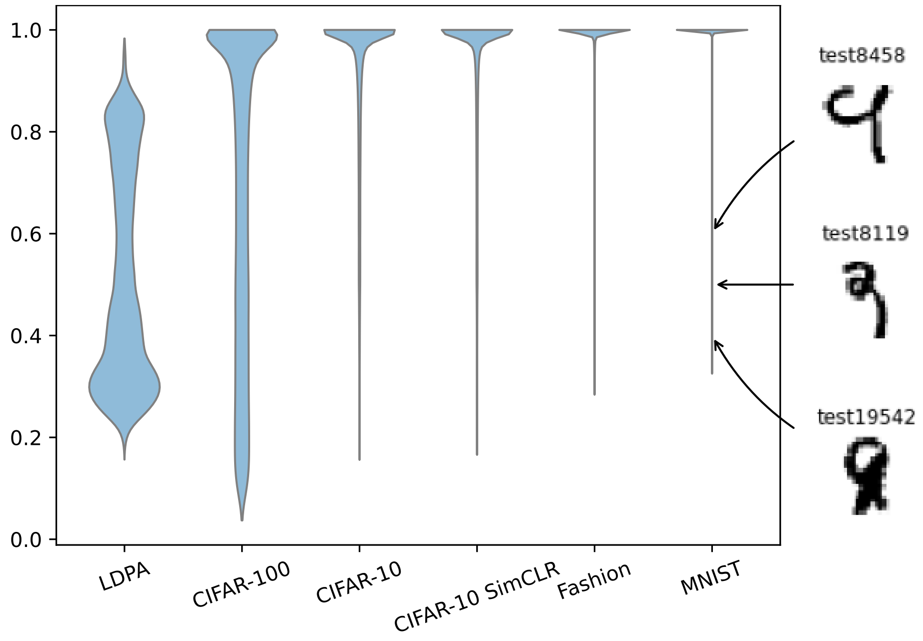


Fig. 1. Distribution of prediction probabilities by a model in a 2-fold setting, and examples of ambiguous samples on the MNIST dataset.

it may also be that this sample is ambiguous and that an ideal classifier would not do any better. The noise is then due to aleatoric uncertainty that cannot be reduced.

Let us call h^{∞} this ideal classifier obtained by training the model on all available training data . We use it to define the theoretical *informed lowest confidence* sampler (denoted by *IConfidence*) based on the following score:

$$\iota(x) = h^{\infty}(x) - h_1^i(x)$$

We expect this sampler to account for aleatoric uncertainty and therefore focus only on reducing epistemic uncertainty. If h^{∞} is unknown at experiment time, it can be estimated in a research context where all labels are known. Such an oracle can be useful in active learning research by providing a golden standard of the maximum achievable accuracy in an experiment.

2.2 Measuring sample noise

Misclassified samples are a source of sample noise, and [5] proposes to identify them using human annotation. This approach can be considered a golden standard but is hard to perform because of human labeling costs.

We previously suggested that sample noise could be measured as the maximum probability predicted by a good enough classifier. In order to extend this measure to a set of samples, we propose to rely on a metric previously introduced in [1] called *reverse batch accuracy* or RBA for short. RBA measures how easy samples are to classify by training a classifier on the test set and measuring its accuracy on sample batches. The lower the RBA score, the harder samples are to classify for the model, so the noisier are the samples.

2.3 Incremental Weighted K-Means (IWKMeans)

The goal of batch active learning strategies is to select batches of samples $\mathcal{D}_B$ representative of the unlabeled data $\mathcal{D}_B \sim \mathcal{D}_U$. For a given notion of similarity sim between batches, this leads to the following maximization objective:

$$\text{argmax}_{\mathcal{D}_B} \text{sim}(\mathcal{D}_B, \mathcal{D}_U) \tag{1}$$

In [8], the similarity is taken as $-\sum_{u \in \mathcal{D}_U} d(\mathcal{D}_B, u)$ with d being the squared distance to the closest point in the set $d(\mathcal{D}_B, u) = \min_{b \in \mathcal{D}_B} \|b - u\|^2$. This corresponds to the inertia objective of the K-Means clustering. The authors propose to use it in a two-step procedure called *Weighted K-Means* (WKMeans) where a set of samples are preselected using margin sampling, and then the final batch is selected by using K-Means.

The above objective does not consider already labeled data and can lead to suboptimal batches lying in regions of high-density of labeled samples. A natural refinement is to additionally impose that the selected batch differs from already labeled data, *i.e.* to minimise similarity $\text{sim}(\mathcal{D}_B, \mathcal{D}_L)$:

$$\text{argmax}_{\mathcal{D}_B} \text{sim}(\mathcal{D}_B, \mathcal{D}_U) \qquad \text{subject to } \text{argmin}_{\mathcal{D}_B} \text{sim}(\mathcal{D}_B, \mathcal{D}_L)$$

In the context of K-Means, minimizing this similarity is equivalent to preventing points close to labeled data to *drag* the centroids toward them. This is done by adding the labeled points in the reference set used to compute distances in the K-Means objective that becomes $-\sum_{u \in \mathcal{D}_U} d(\mathcal{D}_B \cup \mathcal{D}_L, u)$. This translates algorithmically by adding cluster centers corresponding to already labeled samples and keeping them fixed during optimization. We refer to this approach as *Incremental Weighted K-Means* or IWKMeans for short, and it is described in Alg. 1. IWKMeans tends to *repel* batch samples from already selected samples, including the noisy ones. A similar approach is proposed in [3] where the values in the matrix of similarity between batch and selected samples are minimized.

Potential concerns. The fact that the method repels all selected samples and not only the noisy ones can be debated. We tested variants of this method that repels noisy samples only, or noisy and very easy to classify samples as they can also be considered detrimental [1]. Since all variants had similar performances, we present here the simplest one. Another concern is the convergence of this modified version of K-Means. It is easy to imagine in two dimensions how *fixed* centers can prevent a *moving* one to reach its minimum. From our experience, the K-Means++ initialization prevents most of these problems, and Fig. 2 proves

```
Data: $\mathcal{D}_L^0, \mathcal{D}_U^0$
Result: $h^{n_{iter}}$
for $i \leftarrow 1$ to $n_{iter}$ do
    Margin sampling to pre-select $\beta k$ samples among the unlabeled ones:
      $P^i = \arg\max_{\mathcal{D}_U^i} 1 - (h_1^i(x) - h_2^i(x))$
    Perform K-Means on $P^i$ with $k$ moving and $\mathcal{D}_L^i$ fixed centroids:
      $\mathcal{D}_B^i = \arg\min_{\mathcal{D}_B^i \subset P^i} \sum_{x \in P^i} d(\mathcal{D}_B^i \cup \mathcal{D}_L^i, x)$
    Update all sets and train the classifier:
      $\mathcal{D}_L^{i+1} \leftarrow \mathcal{D}_L^i \cup \mathcal{D}_B^i$    $\mathcal{D}_U^{i+1} \leftarrow \mathcal{D}_U^i \setminus \mathcal{D}_B^i$    $h^{i+1} \leftarrow h^i + \mathcal{D}_B^i$
end
```

Algorithm 1: IWKmeans algorithm

the method's efficiency in a two-dimensional setting. For the sake of clarity and concision, we refer the reader to this online study of IWKMeans convergence[1].

3 Experiments

We perform active learning experiments on synthetic and natural datasets following the framework described in [1]. *Random sampling* (Random) is the baseline. We use *KCenterGreedy* (KCenter) as a proxy for Core-sets [6] since there is no open implementation available. Note that the latter uses the activation of the penultimate layer of neural networks, so we have adapted it to random forests by considering a PCA-reduced forest embedding. We compare *lowest confidence sampling* (Confidence) as described above to its informed counterpart *IConfidence.* We also compare *Weighted K-Means*[8] (WKMeans) with $\beta = 10$ to our proposed *IWKMeans*. BatchBALD[4] was not considered due to its prohibitive computational time of several hours compared to less than one minute for others.

We run ten iterations using five repeated two-fold cross-validation for each task. Reported results include means and confidence intervals at 10th and 90th quantiles. Cifar10 and Cifar100 tasks are run on ImageNet embeddings, Cifar10 SimCLR is run on embeddings learned using contrastive learning [2], and other tasks are run using raw data. A Random Forest is used on the LDPA task, all others use a multi-layer perceptron with hidden layers of size 128 and 64. More details can be found on the code repository[2] or in [1].

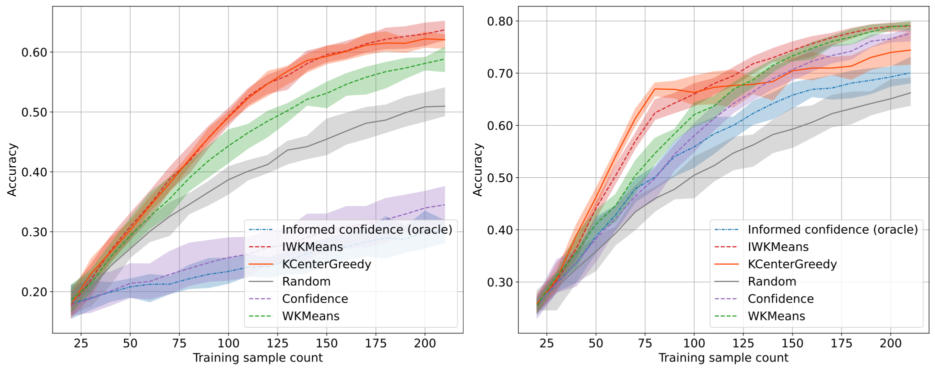
3.1 Synthetic problem with noisy samples

To create noisy samples, we design a task where samples from a given class are not distinguishable from those of another class. We create a 10-class task composed of spatially isolated blobs. Some blobs are composed of regular samples that all belong to the same class. Other blobs are composed of samples randomly

[1] `https://dataiku-research.github.io/cardinal/auto_examples/plot_incr_kmeans.html`

[2] `https://github.com/dataiku-research/paper_ial_2021`

assigned to two different classes; we call them noisy blobs since their samples are impossible to classify. We create a low-dimensional problem with 10000 samples, 2 features, 10 classes, 200 blobs, half of which are noisy. The active learning experiment uses 20 batches of 20 samples. We also create a high-dimensional problem with the same characteristics except that the data has 40 features, and we generate 90 blobs, 30 of which are noisy. We use accuracy AUC over the whole experiment to measure strategy performances. In this synthetic experiment, we know which samples are noisy by construction and therefore report the ratio of noisy samples (NSR) as a measure of sample noise instead of its proxy RBA. Note that RBA is strongly correlated (> 0.95) with NSR. Results are reported in Fig. 2.



(a) 2 features (b) 40 features

Fig. 2. Test accuracy on synthetic problems.

Table 1. AUC and ratio of noisy samples per method. Standard deviation is in parenthesis. Best answers in terms of accuracy (higher) and Noisy Sample Ratio (lower) are in bold.

Dataset	Metric	Random	KCenter	Confidence	IConfidence	WKMeans	IWKMeans
Noisy LD	AUC	38.6 (1.5)	47.9 (0.5)	26.7 (2.5)	24.5 (1.4)	44.0 (1.2)	**48.1** (1.0)
Noisy LD	NSR	50.3 (4.1)	42.4 (2.0)	38.9 (6.5)	**10.1** (4.6)	43.5 (2.6)	39.3 (1.8)
Noisy HD	AUC	50.7 (2.1)	61.7 (1.1)	58.0 (1.2)	55.0 (1.5)	60.6 (0.9)	**63.2** (0.6)
Noisy HD	NSR	35.0 (3.0)	24.5 (1.5)	25.6 (1.5)	**3.2** (1.1)	33.4 (1.5)	26.9 (1.8)

In terms of performances, IWKMeans dominates all methods, which is what was expected. KCenter is closely following which is surprising since the model here is a random forest and we did not expect our quick adaptation to this model to perform well. We would have expected Confidence to select more noisy samples and perform poorly because of that. Instead, it seems to be penalized by its lack of diversity and exploration. IConfidence minimizes the number of noisy samples selected, as expected, and yet it performs as badly as Confidence for the same reasons. In the end, this experiment shows that diversity can be as crucial as sample noise, and we expect a sweet spot to exist. Overall, we also observe that IWKMeans seem to be more robust to noisy samples. More insights are available in appendix Fig. A4.

Table 2. Area under the curve for accuracy (AUC) and reverse batch accuracy (RBA) per method averaged over all repetitions. Standard deviation is in parenthesis. Bold values are statistically significantly higher than the others based on a Friedman test with Nemenyi post-hoc test which details are available in Fig. A5 in appendix.

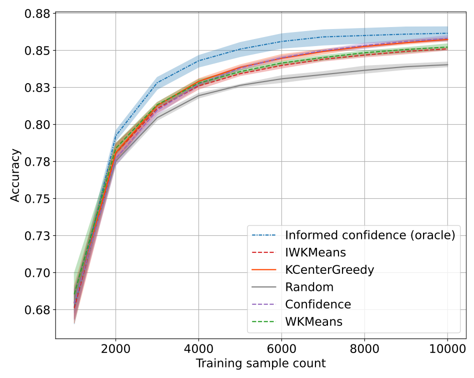
Dataset	Metric	Random	KCenter	Confidence	IConfidence	WKMeans	IWKMeans
LDPA	AUC	**59.0** (0.5)	57.2 (0.5)	51.9 (1.1)	51.2 (0.8)	**63.1** (0.3)	**63.6** (0.3)
LDPA	RBA	67.1 (0.7)	49.3 (2.3)	51.6 (2.0)	**98.9** (0.1)	67.8 (1.1)	67.6 (1.1)
Cifar10	AUC	80.9 (0.2)	**82.0** (0.2)	**81.9** (0.2)	**82.9** (0.4)	81.8 (0.2)	81.6 (0.2)
Cifar10	RBA	91.5 (4.8)	81.5 (10.7)	80.5 (12.6)	**94.9** (3.5)	85.2 (9.0)	85.3 (9.1)
Cifar10S	AUC	88.8 (0.2)	89.2 (0.2)	**89.5** (0.2)	**89.6** (0.3)	**89.4** (0.2)	**89.5** (0.3)
Cifar10S	RBA	93.5 (1.3)	87.5 (1.8)	80.0 (3.6)	**96.5** (0.8)	86.2 (2.8)	87.9 (2.3)
MNIST	AUC	90.9 (0.2)	91.2 (0.3)	**93.5** (0.2)	**93.8** (0.3)	**94.2** (0.1)	**94.2** (0.1)
MNIST	RBA	**97.6** (0.2)	96.6 (0.4)	92.3 (8.1)	**97.7** (2.5)	88.1 (0.4)	86.9 (0.6)
Fashion	AUC	82.4 (0.2)	79.3 (0.3)	**83.5** (0.3)	**85.0** (1.0)	**84.3** (0.1)	**84.3** (0.1)
Fashion	RBA	88.1 (0.4)	**90.8** (9.7)	82.3 (15.9)	**91.3** (7.3)	70.6 (0.7)	69.2 (0.7)
Cifar100	AUC	48.5 (0.3)	**48.3** (0.2)	46.2 (0.2)	**50.8** (0.6)	**48.9** (0.2)	49.0 (0.3)
Cifar100	RBA	69.4 (9.2)	71.2 (14.1)	55.6 (15.6)	**88.8** (5.8)	70.7 (9.2)	70.0 (9.9)

3.2 Real datasets

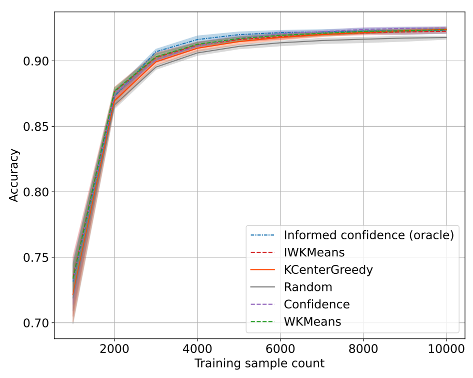
We now analyze the samplers behaviors on our collection of real-life datasets.

Informed lowest confidence. IConfidence is equivalent or better than confidence in all cases. It is also the best strategy for all tasks except MNIST and LDPA. Note that the RBA of this method is much higher than the other strategies. It reveals that getting *too close* to the decision boundary may not be required for good performance. Even more, this oracle method does not enforce diversity but yet overpowers diversity enforcing methods. This questions the fundamental hypothesis that enforcing diversity is mandatory to obtain good performances. Further work will investigate further this sampler and try to reproduce its behavior online with proxy metrics proposed in [1].

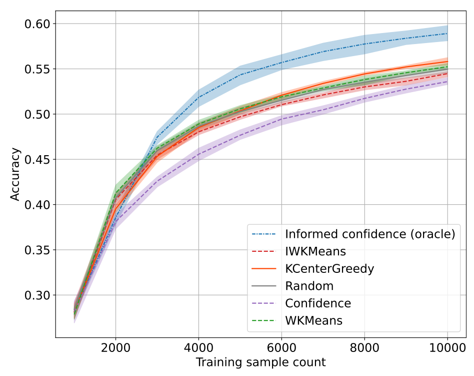
IWKMeans. WKMeans and IWKMeans bring a significant uplift against random and all other uncertainty-based or unsupervised methods in all tasks except CIFAR10 with SimCLR embeddings. IWKMeans outperforms WKMeans on LDPA only, making it hard to draw a definitive conclusion on real tasks. Further experiments are needed to investigate these behaviors. Early investigations suggest that the variation in density of noisy samples in multiclass settings



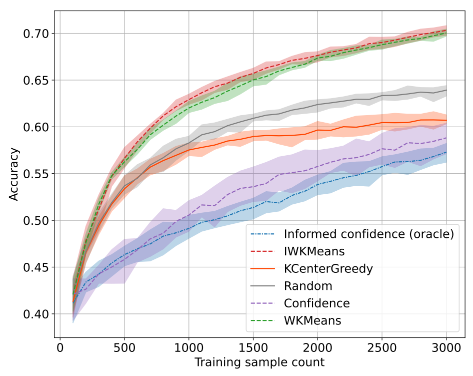
(a) CIFAR 10, ImageNet embedding



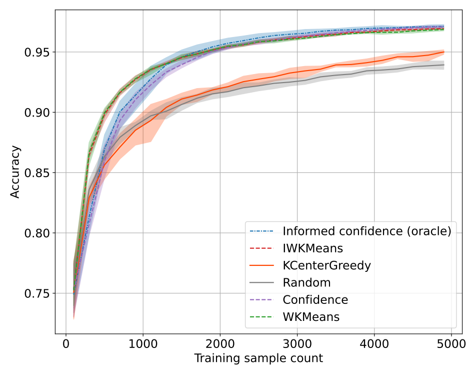
(b) CIFAR 10, SimCLR embedding



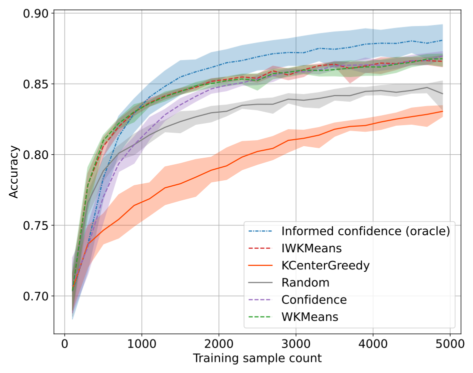
(c) CIFAR 100



(d) LDPA



(e) MNIST



(f) Fashion MNIST

Fig. 3. Results on real datasets

can tamper with adverse-to-noise samplers. For example, a general strategy can be hard to find on the MNIST dataset where few noisy samples exist between classes zero and four, while their density is high between classes three and five.

SimCLR embedding. An unexpected conclusion of these experiments is that contrastive-based embeddings can bring an uplift significantly higher than choosing the best query sampling strategy.

4 Conclusion

In active learning, noisy samples that are hard to classify by the model can be detrimental to the performance. To prove this, we have designed a metric to measure them and a synthetic problem to test the robustness of query strategies to their presence. IWKmeans, the proposed noise-adverse sampling strategy, has been proven effective on synthetic data, but not on real tasks where it marginally improves WKMeans on which it is based. If IWKMeans' performance seems correlated to the number of noisy samples selected, there may be more than meets the eye in this problem, and more investigations are needed. Our study also shows that a sampler as simple as confidence sampling can outperform all other samplers if informed by a good enough classifier. Whether or not this uplift can be reproduced in real conditions using a proxy must be investigated in further work.

References

[1] Alexandre Abraham and Léo Dreyfus-Schmidt. "Rebuilding Trust in Active Learning with Actionable Metrics". In: *2020 IEEE International Conference on Data Mining Workshops (ICDMW)* (2020).

[2] Ting Chen et al. "A simple framework for contrastive learning of visual representations". In: *International conference on machine learning*. PMLR. 2020, pp. 1597–1607.

[3] Bo Du et al. "Exploring representativeness and informativeness for active learning". In: *IEEE transactions on cybernetics* 47.1 (2015), pp. 14–26.

[4] Andreas Kirsch, Joost Van Amersfoort, and Yarin Gal. "Batchbald: Efficient and diverse batch acquisition for deep bayesian active learning". In: *arXiv preprint arXiv:1906.08158* (2019).

[5] Curtis G Northcutt, Anish Athalye, and Jonas Mueller. "Pervasive label errors in test sets destabilize machine learning benchmarks". In: *arXiv preprint arXiv:2103.14749* (2021).

[6] Ozan Sener and Silvio Savarese. "Active learning for convolutional neural networks: A core-set approach". In: *arXiv preprint arXiv:1708.00489* (2017).

[7] Burr Settles. *Active learning literature survey*. Tech. rep. Department of Computer Sciences, University of Wisconsin-Madison, 2009.

[8] Fedor Zhdanov. "Diverse mini-batch Active Learning". In: *arXiv preprint arXiv:1901.05954* (2019).

Contrastive Representations for Label Noise Require Fine-Tuning

Pierre Nodet[1,2], Vincent Lemaire[1], Alexis Bondu[1], and Antoine Cornuéjols[2]

[1] Orange Labs, Paris & Lannion, France
[2] AgroParisTech, Paris, France

Abstract. In this paper we show that the combination of a Contrastive representation with a label noise-robust classification head requires fine-tuning the representation in order to achieve state-of-the-art performances. Since fine-tuned representations are shown to outperform frozen ones, one can conclude that noise-robust classification heads are indeed able to promote meaningful representations if provided with a suitable starting point. Experiments are conducted to draw a comprehensive picture of performances by featuring six methods and nine noise instances of three different kinds (none, symmetric, and asymmetric). In presence of noise the experiments show that fine tuning of Contrastive representation allows the six methods to achieve better results than end-to-end learning and represent a new reference compare to the recent state of art. Results are also remarkable stable versus the noise level.

1 Introduction

Deep Learning (DL) paradigm has proved very powerful in many tasks, however recent papers [34, 49] have shown that **"noisy labels"** are a real challenge for end-to-end deep learning architectures. Their test performance is found to deteriorate significantly even if they are able to learn perfectly the train examples. This problem has attracted a lot of suggestion in many recent papers.

Zhang et al. [50] conducted experiments to analyze the impact of label noise on deep architectures, and they found that the performance degradation mainly comes from the representation learning rather than the classification part. It therefore appears very difficult to learn a relevant representation in the presence of label noise, in an end-to-end manner.

To tackle this problem, one option is to exploit an already existing representation which has been learned in an unsupervised way. In particular, Self Supervised Learning [24] (SSL) gathers an ensemble of algorithms which automatically generate supervised tasks from unlabeled data, and, therefore to learn representations from examples that are not affected by label noise. An example of SSL algorithm is Contrastive Learning [21], where a representation of the data is learned by making feature vectors from similar pictures (i.e. generated from the same original picture by using two different transformer functions) to be close in the feature space whereas feature vectors from dissimilar pictures are to be far apart. In [13], the authors propose to initialize the representation with a pre-trained Contrastive Learning one, and then, to use the noisy labels to learn the classification part and fine-tune the representation. It appears that this approach clearly outperforms the end-to-end architecture, where the representation is learned from noisy labels.

But questions remain: is this performance improvement only attributable to the quality of the Contrastive Representation used (i.e. the starting point of fine-tuning)? Or is the fine-tuning step able to promote a better representation? To answer these questions this paper examines the different possibilities to learn a DL architecture in presence of label noise: (i) end-to-end learning (ii) learning only the head part when freezing a contrastive representation and (iii) fine tuning the later representation.

The rest of this paper is organized as follows. The section 2 provides a brief overview of the main families of algorithms dedicated to fight the label noise underlying the issue of preserving a good representation in spite of label noise. Section 3 then describes the experimental protocol. The section 4 will present the results and a deep analysis which will allow us to answer the questions above. The last section raises an interesting conclusion and provides some perspectives for future work.

2 Representation Preserving with Noisy Labels

This section presents a brief overview of the state of the art on *learning deep architecture with noisy labels* emphasizing how these methods *preserve*, to some extent, *the learned representation* in the presence of label noise. For an extended overview, the reader may look [43].

2.1 Preserving by Recovering

The dominant approach to preserve the learned representation is to recover a clean distribution of the data from the noisy dataset. It mostly consists in finding a mapping function from the noisy to the clean distribution thanks to heuristics, algorithms or machine learning models. Three different ways of recovering the clean distribution are usually put forward [36]: (i) sample reweighting; (ii) label correction and (iii) instance moving.

Recovering by Reweighting - The sample reweighting methodology aims at assigning a weight to every samples such that the reweighted population behaves as being sampled from the clean distribution. The Radon-Nikodym derivative (RND) [35] of the clean concept with respect to the noisy concept is the function that defines the perfect reweighting scheme. Many algorithms therefore rely on providing a good estimation of the RND by learning it from the data using Meta Learning [38] or minimizing the Maximum Mean Discrepancy of both distributions in a Reproducing kernel Hilbert space [8, 32]. Many of these methods are inspired by the covariate shift problem [14, 20]. Other algorithms rely on different reweighting schemes that do not involve the RND as done, for instance, in Curriculum Learning [2]. They are described in details later in this section. By doing sample reweighting, algorithms evaluate whether or not a sample is deemed to have been corrupted and assign a lower weight to a suspect sample so that its influence on the training procedure is lowered. The hope is that clean samples are sufficient to learn high-quality representations

Recovering by Relabelling - Another way to recover the clean distribution from the noisy data is to correct the noisy labels. One great advantage over sample reweighting is that corrected samples can be fully used during the training procedure. Indeed, when a sample is corrected, it will count as one entire sample in the training procedure (gradient descent for example), whereas a reweighted noisy sample would get a low weight and would not be used significantly in the training procedure. Thus, when done effectively, label correcting might get better performance. Meta Label Correction (MLC) [42] is an example of this approach where the label correction is done thanks to a model learned using meta learning. One downside of label correction, however, is that the label of a clean sample can get "corrected" or the label of a noisy sample can get changed to a wrong label. Label correcting algorithm assign the same weight to all training examples, even though they might have "corrected" a label based on shaky assumptions. By contrast, Sample reweighting will assign a low weight if the algorithm is not confident in whether the sample is clean or noisy.

Recovering by Modifying - A third way to recover the clean distribution is by modifying the sample itself so that its position in the feature space gets closer or is moved within an area for its label that seems more appropriate (i.e. obeying regularisation criteria). Finding a transformation in the latent space itself has the advantage to require less labelled samples, or even none at all, as the work is performed on distance between samples themselves, like for example in [40].

2.2 Preserving by Collaboration

Multiple algorithms and agreements measures have been used in many sub-fields of machine learning such as ensembling [4,10,11] or semi supervised learning [3,48]. They can be adapted to learn with noisy labels by relying on a disagreement method between models in order to detect noisy samples. When the learned models disagree on predictions for the label of a sample, this is considered as a sign that the label of this sample may be noisy. When the models used are diverse enough, these methods are often found to be quite efficient [17,46].

However these algorithms suffer from learning their own biases and diversity needs to be introduced in the learning procedure. Using algorithms from different classes of models and different origins can increase the diversity among them by introducing more source of biases [28]. Alternating between learning from the data and from the other models is another way to combat the reinforcement of the models' biases [46]. These algorithms rely on carefully made heuristics to be efficient.

2.3 Preserving by Correcting

When learning loss base models, such as neural networks, on label noise, the loss value of a training example can be a discriminative feature to decide if its label is noisy. Deep neural networks seem to have the property that they first learn general and high level patterns from the data before falling prey to overfitting the training samples, especially in the presence of noisy labels [1, 30]. As they are "learned" at a later stage, these noisy examples are often associated with a high loss value [16] which may then highly influences the training procedure and perturb the learned representation [49]. A way to combat label noise is accordingly to focus first on small loss and easy examples and keep the high loss and hard examples for the end of the training procedure. Curriculum Learning [2] is a way to employ this training schema with heuristic based schemes [9, 22, 27, 31] or schemes learned from data [23, 41]. This class of algorithm has the same properties as the ones relying on importance reweighting, but maybe more adapted to training with iterative loss based algorithms such as neural networks or linear models.

Instead of filtering or reweighting samples based on their loss values, one could try to correct the loss for these samples using the underlying noise pattern. Numerous method have been doing so by estimating the noise transition matrix for *Completely at Random* (i.e uniform) and *At Random* (i.e class dependent) noise [19, 37, 42]. This category of algorithms are still to be tested on more complex noises scenarios such as *Not at Random* (i.e instance and class dependent) noise.

2.4 Preserving by Robustness

The last identified way to preserve the learned representation of a deep neural network in presence of label noise is by using a robust or regularized training procedure. This can take multiple forms from losses to architectures or even optimizers. One of them are Symmetric Losses [5,12,39]. A symmetric loss has the property that: $\forall x \in \mathcal{X}$, $\sum_{y \in \mathcal{Y}} L(f(x), y) = c$ where $c \in \mathbb{R}$. These losses have been proven to be theoretically insensitive to Completely at Random (CAR) label noise. Recently, modified versions of the well-known Categorical Cross Entropy (CCE) loss have been designed in order to be more robust and thus more resistant to CAR label noise as is the case for the Symmetric Cross Entropy (SCE) loss [45] or the Generalized Cross Entropy Loss (GCE) [51]. Both of these rely on using the CCE loss combined with a known more robust loss such as the Mean Absolute Error (MAE). However, the resulting algorithms often underfit in presence of too few label noise while they are unable to learn a correct classifier with too much label noise.

All these approaches still adopt the end-to-end learning framework, aiming at fighting the effects of label noise by preserving the learned representation. However they fail to do so in practice: *decoupling* the learning of the representation, using Self Supervised (SSL) learning, from the classification learning stage itself and then fine tuning the representation with robust algorithms is beneficial for the model performance [13, 50]. A natural question arises about the origin of the performance improvements, and the ability of these algorithms to learn or promote a good representation in presence of label noise. If robust algorithms are unable to learn a representation it should be even better to freeze the SSL representation instead of fine tuning it.

In order to assess the origin of the improvements for different classes of algorithms and different noise levels, we compare the above-mentioned end-to-end approaches against each other when the representation is learned in a self-supervised fashion by either fine tuning or freezing the representation when the classification head is learned. Thus, any difference in the performance would be attributable to the difference in the representation learnt.

3 Experimental Protocol

In [50], the authors showed that when using end-to-end learning, fine tuning the representation on noisy labels harms a lot the final performance, while learning a classifier on frozen embeddings is quite robust to label noise and leads to significant performance improvements over state-of-the-art algorithms if the representation is learned using trustful examples. The latter can be found for instance using confidence and loss value. Nonetheless it is arguable whether these improvements were brought by an efficient self-supervised pretraining (SSL) with SimCLR [6], a contrastive learning method, or by the classification stage of the REED algorithm [50].

The goal of the following experimental protocol is to assess and isolate the role of the contrastive learning stage, in the performance that can be achieved by representative methods as presented in Section 2 about state of the art approaches. Specifically, several RLL algorithms have been chosen, one from each of the highlighted families (see Section 2 and Table 1). For each, the difference in performance between using contrastive learning to learn the representation and the performance reported with the original end-to-end algorithms is measured. These experiments seek to highlight the impact of each RLL algorithms and assess if these are able to promote a better representation than the pretrained contrastive representation through fine-tuning.

The rest of this section describes the experimental protocol used to conduct this set of experiments.

3.1 The tested Algorithms

Section 2 presented an overview of the state of the art for learning with label noise organized around families of approaches that we highlighted. Since our experiments aim at studying the properties of each of these approaches, we selected one representative technique from each of these families as indicated in the following.

– In the first family of techniques (*recover the clean distribution*), the algorithms re-weight the noisy examples or attempt to correct their label. One of these algorithm uses what is called Dynamic Importance Reweigthting **(DIW)**. It reweights samples using Kernel Mean Matching (KMM) [14, 20] as is done in covariate shift with Density Ratio Estimators [44]. Because this algorithm adapts well-grounded principles to end-to-end deep learning, it is a particularly relevant algorithm for our experiments.

– CoLearning **(CoL)** [46] is a good representative of the family of *collaborative learning algorithms*. It uses disagreements criteria to detect noisy labels and is tailored for end-to-end deep learning where the two models are branches of a larger neural networks. It appears to be one of the best performing collaborative algorithm while not resorting to complex methods such as data augmentation or probabilistic modelling like the better known DivideMix [29].
– The third identified way to combat label noise is by *mitigating the effect of high loss samples* [16] by either ditching them or using a loss correction approach. Curriculum learning is often used to remove the examples that are associated with high loss from the training set. **(MWNet)** [41] is one the most recent approach using this technique, which learns the curriculum from the data with meta learning. Besides, Forward Loss Correction **(F-Correction)** [37] and Gold Loss Correction **(GLC)** [19] are two of the most popular approaches to combat label noise by *correcting the loss function*. Both seek to estimate the transition matrix between the noisy labels to the clean labels, the first technique using a supervised approach thanks to a clean validation set, and the second one in an unsupervised manner. Even though many extensions of these algorithm have been developed since then [42,47], in these experiments, we use F-Correction and GLC since they are way simpler and almost as effective.
– Lastly, in recent literature, a new emphasis is put on the research of new loss functions that are conducive to better risk minimization in presence of noisy labels *for robustness purpose*. For example, [5, 39] show theoretically and experimentally that when the loss function satisfies a symmetry condition, described below, this contributes to the robustness of the classifier. The Generalized Cross Entropy **(GCE)** [51] is the robust loss chosen in this benchmark as it appears to be very effective.

A note *about additional requirements*: These algorithms may have additional requirements, mostly some knowledge about the noise properties. These are described in table 1. In the experiments presented below, the clean validation dataset is set to be 2 percent of the total training data, like in [41, 53], and the noise probability is provided to the algorithms that need it.

A note *about the choice of the pretrained architecture*: We chose to use SimCLR for Self-Supervised Learning (SSL) as done in [50].

SimCLR is a contrastive learning algorithm that is composed of three main components (See Figure 1): a family of data augmentation $\mathcal{T}$, an encoder network $f(\cdot)$ and a projection head $g(\cdot)$. Data augmentation is used as a mean to generate positive pairs of samples: a single image $\mathbf{x}$ is transformed into two similar images

Algorithms (Date)	**Noise Ratio**	**Clean Validation**	**Family (Section)**
DIW (2020)	×	✓	Reweighting (2.1)
CoLearning (2020)	✓	×	Collaborative Learning (2.2)
MWNet (2019)	×	✓	Curriculum Learning (2.3)
F-Correction (2017)	×	×	Loss Correction (2.3)
GLC (2018)	×	✓	Loss Correction (2.3)
GCE (2018)	×	×	Robust Loss (2.4)

Table 1. Taxonomy of robust deep learning algorithms studied in this paper. The **Noise Ratio** column corresponds to whether the algorithm needs the noise rate (✓) to learn from noisy data or not (×). The **Clean Validation** column corresponds to whether the algorithm needs an additional clean validation dataset (✓) to learn from noisy data or not (×).

$\tilde{\mathbf{x}}_i$ and $\tilde{\mathbf{x}}_j$ by using a data augmentation module $\mathcal{T}$ with different seeds t and t'. Then the two images go through an encoder network $f(\cdot)$ to extract an image representation $\mathbf{h}$, such as $\mathbf{h}_i = f(\tilde{\mathbf{x}}_i)$ and $\mathbf{h}_j = f(\tilde{\mathbf{x}}_j)$. Finally a projection head $g(.)$ is used to train the contrastive objective in a smaller sample space $\mathbf{z}$, with $\mathbf{z}_i = g(\tilde{\mathbf{h}}_i)$ and $\mathbf{z}_j = g(\tilde{\mathbf{h}}_j)$. The contrastive loss used is called the NT-Xent, the normalized temperature-scaled cross entropy loss, and defined by the following formula:

$$\ell(\mathbf{z}_i, \mathbf{z}_j) = -\log \frac{\exp(\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_j)/\tau)}{\sum_{k=1}^{2N} \exp(\mathrm{sim}(\mathbf{z}_i, \mathbf{z}_k)/\tau)} \tag{1}$$

where τ is the temperature scaling and sim is the cosine similarity. The final loss is computed across all positive pairs, both (i, j) and (j, i), in a mini-batch. When the training of SimCLR is complete, the projection head $g(.)$ is dropped and the embeddings $\mathbf{h}$ are used as an image representation in downstream tasks.

Other SSL algorithms could have been used as well, such as Moco [7,18] or Bootstrap Your Own Latent (BYOL) [15]. However, we do not expect that the main conclusions of the study would be much changed.

3.2 Datasets

The datasets chosen in this benchmark are two image classification datasets namely CIFAR10, CIFAR100. They are two famous image classification datasets, containing only clean examples and as such, we will simulate symmetric (Complelty at Random) and asymmetric (At Random) noise as defined later in section 3.3. These benchmarks should be extended to other image classification datasets such as FashionMNIST, Food-101N, Clothing1M and Webvision and to other classification tasks such as text classification or time series classification.

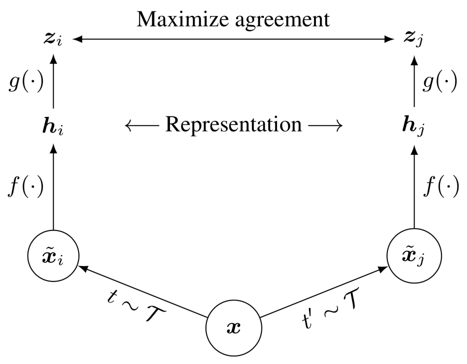


Fig. 1. Figure from [6]: "A simple framework for contrastive learning of visual representations. Two separate data augmentation operators are sampled from the same family of augmentations ($t \sim \mathcal{T}$ and $t' \sim \mathcal{T}$) and applied to each data example to obtain two correlated views. A base encoder network $f(\cdot)$ and a projection head $g(\cdot)$ are trained to maximize agreement using a contrastive loss. After training is completed, we throw away the projection head $g(\cdot)$ and use encoder $f(\cdot)$ and representation $\mathbf{h}$ for downstream tasks."

3.3 Simulated Noise

As datasets chosen in Section 3.2 contains clean labels, label noise will be introduced synthetically on the training samples. Two artificial noise models will be used, a symmetric (Completely at Random) and asymmetric (At Random) noise. Symmetric noise corrupts a label from one class to any other classes with the same probability, meanwhile the asymmetric corrupts a label to a similar class only. Similar classes are defined through class mappings. For CIFAR-10, the class mappings are TRUCK $\rightarrow$ AUTOMOBILE, BIRD$\rightarrow$ AIRPLANE, DEER $\rightarrow$ HORSE, CAT $\leftrightarrow$ DOG. For CIFAR-100, the class mappings are generated from the next class in that group (where 100 classes are categorized into 20 superclasses of 5 classes). These class mappings are the ones introduced in [37, 51].

3.4 Implementation Details

We give some implementation details for reproducibility and / or a better understanding of the freezing process in the experiments:

– On CIFAR10 and CIFAR100 the SGD optimizer will be used to train the final Multinomial Logistic Regression with an initial learning rate of 0.01, a weight decay of $1e^{-4}$ and a non-Nesterov momentum of 0.9. The learning rate will be modified during training with cosine annealing [33]. The batch size is 128.
– When doing the *"Freeze"* experiments, the weights of SimCLR from [13] will be used and will not be modified during the training procedure. All

the weights up to before the projection head of SimCLR are used, then the dimension output of the feature encoder is 2048 for CIFAR10 and CIFAR100. The classification architecture is composed by a single linear layer with an output dimension of 10 (or 100), corresponding to the number of classes. Thus when trained with the Categorical Cross Entropy it corresponds to a usual logistic regression. This classifier is going to be learned with multiple algorithms robust to label noise. These algorithms are not modified from their original formulation.
– The *"Fine Tuning"* experiments follow the same implementation as the *"Freeze"* experiments. However the weights of the same pretrained SimCLR encoder are allowed to be modified by backpropagation.
– Based on their public implementation and / or article we re-implemented all the algorithm tested (DIW [8], CoL [46], MWNet [41], F-Correction [37], GLC [19] and GCE [51]). All these re-implemented algorithms will soon be available as an open source library easily usable by researchers and practitioners. These custom implementations have been verified to produce, under the same condition stated in the corresponding original papers (noise models, network architectures, optimizers, ...), the same results or results in the interval of confidence (for clean or noisy labels). We may thus be confident that results in the different parts of the Tables 2 and 3 are comparable.
– The experiments have been run multiple times for all algorithms, some datasets, some noise models and some noise ratios with different seeds to see the seed impact on the final performance of the classifier. For all algorithms, the standard deviation of the accuracy was less than 0.1 percent.

4 Results

This section reports the results obtained using the protocol described in section 3. They are presented in the tables 2 and 3 corresponding to the two tested datasets CIFAR10 and CIFAR100. Each table is composed of four rows subsections corresponding to the different types of representation used, which can be learned in a *End-to-End* manner (A), be taken from an already existing SSL model, either *Frozen* (B) or *Fine tuned* (C). Moreover they are composed of two columns subsections corresponding to the noise model used to corrupt samples (symmetric or asymmetric).

These tables present the results from different studies: (A) The first part of these tables about *"End-to-End learning"* are results reported in the respective papers [8, 19, 37, 41, 46, 51] or reported in [13]; (B) The second part about *"Freeze"* experiments conducted in this paper, are made by re-implementing the referred algorithms from scratch; (C) The *"Fine Tuning"* experiments are results reported in [13].

The interpretation of the Table 2 and 3 will be done in two times, first a comparison between whole blocks (as (A) against (B)) will give insights on how deep neural networks learn representations on noisy data and how robust algorithms helps to improve the learning process or helps to preserve a given

Algorithms		CIFAR10								
		Clean	Symmetric						Asymmetric	
		0	20	40	60	80	90	95	20	40
DIW [8]	End-to-End (A)			80.4	76.3					84.4
CoL [46]			93.3	91.2		49.2			88.2	82.9
MWNet [41]		95.6	92.4	89.3	84.1	69.6	25.8	18.5	93.1	89.7
F-Correction [37]		90.5	87.9			63.3	42.9		90.1	
GLC [19]		95.0	95.0	95.0	95.0	90.0	80.0	76.0		
GCE [51]		93.3	89.8	87.1	82.5	64.1			89.3	76.7
DIW	Freeze (B)	91.3	91.2	90.8	90.5	89.8	89.2	88.1	91.0	90.6
CoL		91.1	91.1	90.9	90.6	89.9	89.4	88.8	90.8	89.9
MWNet		91.3	91.2	90.8	90.6	89.8	88.2	82.4	90.9	86.4
F-Correction		90.8	90.5	90.1	89.6	88.4	88.0	88.1	88.9	88.4
GLC		90.7	89.7	90.0	89.5	89.0	88.5	88.3	88.7	88.2
GCE		91.1	90.8	90.7	90.5	90.4	90.0	89.1	90.9	89.0
DIW	Fine Tuning (C)	94.5	94.5	94.5	94.5	94.0	92.0	89.1	94.2	93.6
CoL		93.9	94.6	94.6	94.2	93.6	92.7	91.7	94.0	93.7
MWNet [13]		94.6	93.9	92.9		91.5	90.2	87.2	93.7	92.6
F-Correction		94.0	93.4	93.1	92.9	92.3	91.4	90.0	93.6	92.8
GLC		93.5	93.4	93.5	93.1	92.0	91.2	88.3	93.2	92.1
GCE [13]		94.6	94.0	92.9		90.8	88.4	83.8	93.5	90.3

Table 2. Final accuracy for the different models on CIFAR10 under symmetric and asymmetric noises and multiple noise rates.

representation. Then in a second time comparisons in a given block will be made against multiple algorithms to see how well these conclusions works on different preservation families given in Section 2.

First, we observe when comparing section (A) and (B) from both tables that *"Freeze"* experiments consistently outperforms *"End-to-End"* experiments as soon as the data stop being perfectly clean. Using a pretrained self-supervised representation such as SimCLR improves significantly the performances of the final classifier. Outside of well controlled and perfectly clean datasets all selected algorithms are not able to learn a good enough representation from the noisy data and are beaten by a representation learned without resorting to using given labels. Robust Learning to Label noise algorithms, especially designed for deep learning, can preserve an already good representation from noisy labels but are unable to learn a good representation from scratch.

Then, we observe when comparing section (B) and (C) from both tables that *"Fine Tuning"* experiments consistently outperforms *"Freeze"* at noise rates less than 80 for the symmetric case and less than 40 for the asymmetric case. The nature of the final classifier used after the learned representation partially explains these results; we used a single dense layer (see Section 3.4). This classifier may under-fit as the number of learnable parameters might be too low to actually fit complex datasets such as CIFAR10 and CIFAR100 even with a good given representation. Using more complex classifiers such as Multi-Layer Perceptron could have led to comparable performances than fine tuning even for low noise rates. This point leaves room for further investigation. Having the possibility to

Algorithms		CIFAR100								
		Clean	Symmetric						Asymmetric	
		0	20	40	60	80	90	95	20	40
DIW [8]	End-to-End (A)			53.7	49.1					54.0
CoL [46]			75.8	73.0		32.8				
MWNet [41]		79.9	74.0	67.7	58.7	30.5	5.2	3.0	71.5	56.0
F-Correction [37]		68.1	58.6			19.9	10.2		64.2	
GLC [19]		75.0	75.0	75.0	62.0	44.0	24.0	12.0	75.0	75.0
GCE [51]		76.8	66.8	61.8	53.2	29.2			66.6	47.2
DIW	Freeze (B)	65.6	65.1	64.0	62.9	59.0	53.3	42.5	61.7	49.0
CoL		65.8	65.0	64.0	63.4	62.3	60.0	57.0	64.1	58.6
MWNet		66.6	66.6	66.2	65.4	63.7	59.8	49.5	64.8	54.5
F-Correction		66.5	64.7	61.8	58.8	54.5	51.7	50.8	58.4	56.5
GLC		58.5	57.8	52.3	51.1	41.6	40.1	35.3	51.4	50.3
GCE		63.5	62.9	61.5	60.0	55.7	51.0	49.9	51.2	48.3
DIW	Fine Tuning (C)	73.8	74.9	74.9	74.5	70.2	62.3	50.4	71.8	62.8
CoL		73.7	74.8	74.8	75.0	73.2	67.3	62.0	72.6	70.3
MWNet [13]		75.4	73.2	69.9		64.0	57.6	44.9	72.2	64.9
F-Correction		69.8	70.1	69.1	69.5	66.9	62.1	57.0	70.3	66.2
GLC		69.7	69.4	68.6	62.5	50.4	32.1	18.7	68.2	62.3
GCE [13]		75.4	73.3	70.1		63.3	55.9	45.7	71.3	59.3

Table 3. Final accuracy for the different models on CIFAR100 under symmetric and asymmetric noises and multiple noise rates.

fine tune the representation to better fit the classification task induces the risk to actually degrade it.

Outside of well controlled and perfectly clean datasets, practitioners should first consider to learn a self-supervised representation and then either fine tune it or freeze it with classifier learned with robust algorithms. Self-Supervised Learning (SSL) algorithm such as SimCLR seems to perfectly fit this task, but other SSL algorithms could be used and explored.

Another observation from this benchmark is about the difference in performance between all the tested algorithms. Indeed, if we consider part (B) of Table 2, for both noise models and all noise rates, the performances between the algorithms are close, around 0.1 point in accuracy with some exceptional data points. It shows that even complex algorithms have a hard time beating simpler approaches when they are compared with an already learned representation.

The same observation can be done for the part (B) of Table 3 (for CIFAR 100), especially for the symmetric noise. However the differences between algorithms are better put in perspective with this more complex dataset which contains 10 time more classes and 10 time less samples per classes. We notice that some algorithms start to struggle at high symmetric noise rate or for the more complex asymmetric noise model. For example, GLC is under-performing against competitors for all cases and is under-performing against its end-to-end version. One reason could be the small size used for the validation dataset as the transition matrix is evaluated on it in a supervised manner. The small number of samples may impact the performance of the transition matrix estimator. Much less so than the estimator proposed by F-Correction which seems to perform fine even on CIFAR100 for all symmetric noises, yet only above average on asymmetric noises. Seeing F-Correction and GLC not performing well on asymmetric noise for both dataset is surprising as these algorithms were both particularly designed for this case.

Lastly we observe on both Tables 2 and 3 that algorithms with additional knowledge on the noise model (see Table 1) have an edge over algorithms that do not, especially on the hardest cases with more classes, higher noise ratio or more complex noise model. CoL requires the noise ratio as its efficiency relies on the hyper parameters value corresponding to the injection of pseudo labels and confidence in model prediction that are dependent of the noise ratio. CoL emerges among the most well rounded and most efficient algorithm for all noise models, noise rates and datasets thanks partially to this additional knowledge. On the other hand, GLC, DIW and MWNet require an additional clean validation dataset in order to estimate the noise model or a proxy of it to correct the learning procedure on the noisy dataset. We could expect these algorithms to perform better than CoL as they would be able to deal with more complex noise models and have a fine-grained policy for correcting noisy samples. Still these algorithms are not able in these experiments to get a better accuracy than CoL and perform on par with it.

Finally we need to emphasize that only two datasets have been used in this study, specially two datasets about image classification. In order to stronger our claims, more experiments should be conducted.

5 Conclusion

In this paper our contribution was to suggest new insights about *decoupling* against *end-to-end* deep learning architectures to learn, preserve or promote a good representation in case of label noise. We presented (i) a new view on a part of the state of the art: the ways to preserve the representation (ii) and an empirical study which completes the results and the conclusions of other recent papers [13, 50, 52]. Experiments conducted draw a comprehensive picture of performances by featuring six methods and nine noise instances of three different kinds (none, symmetric, and asymmetric). Our added value for the empirical study is the comparison between the "freeze" and the "fine tuning" results.

One conclusion we are able to draw is that designing algorithms that preserve or promote good representation under label noise is not the same as designing algorithms capable of learning from scratch a good representation under label noise. To make end-to-end learning succeed in this setup researchers should take a better approach when designing such algorithms.

Another element that emerged from the experiments was the efficiency of both freeze and fine tuning approaches in comparison to the end-to-end learning approach. Even the most complex algorithms such as DIW when trained in an end-to-end manner are not able to beat simple robust loss as GCE when trained with fine tuning. It questions usual experimental protocols of Robust Learning to Label (RLL) noise papers and questions the recent advances in the field.

Evaluating RLL algorithms with pretrained architectures should be the norm as it is easy to do so and the most efficient way for practitioners to train model on noisy data.

One more strong point in this conclusion is that in presence of noise the experiments show that fine tuning of Contrastive representation allows the six methods to achieve better results than their end-to-end learning version and represent a new reference compare to the recent state of art. Results are also remarkable stable versus the noise level.

Since fine-tuned representations are shown to outperform frozen ones, one can conclude that noise-robust classification heads are indeed able to promote meaningful representations if provided with a suitable starting point (contrastingly to readers of [13, 52] who might prematurely jump to the inverse conclusion).

However these experiments could be extended to be more exhaustive in two ways: (i) SimCLR is not the only recent and efficient contrastive learning algorithms, MOCO [7, 18] or Bootstrap Your Own Latent (BYOL) [15] could have been used as said earlier in the paper, but other self-supervised or unsupervised algorithms could have been used such as Auto-Encoder [26] or Flow [25]; (ii) experiments could be extended to datasets from other domains such as text classification or time series classification.

Acknowledgements

We would like to thank the anonymous reviewers for their careful, valuable and constructive reviews as well as the words of encouragement on our manuscript.

References

1. Arpit, D., Jastrzebski, S., Ballas, N., Krueger, D., Bengio, E., Kanwal, M.S., Maharaj, T., Fischer, A., Courville, A., Bengio, Y., Lacoste-Julien, S.: A closer look at memorization in deep networks. In: International Conference on Machine Learning. pp. 233–242 (2017)
2. Bengio, Y., Louradour, J., Collobert, R., Weston, J.: Curriculum learning. In: International Conference on Machine Learning. pp. 41–48 (2009)
3. Blum, A., Mitchell, T.: Combining labeled and unlabeled data with co-training. In: Proceedings of the eleventh annual conference on Computational learning theory. pp. 92–100 (1998)
4. Breiman, L.: Bagging predictors. Machine Language **24**(2), 123–140 (Aug 1996)
5. Charoenphakdee, N., Lee, J., Sugiyama, M.: On symmetric losses for learning from corrupted labels. In: International Conference on Machine Learning. vol. 97, pp. 961–970 (2019)
6. Chen, T., Kornblith, S., Norouzi, M., Hinton, G.: A simple framework for contrastive learning of visual representations. In: III, H.D., Singh, A. (eds.) Proceedings of the 37th International Conference on Machine Learning. Proceedings of Machine Learning Research, vol. 119, pp. 1597–1607. PMLR (13–18 Jul 2020)
7. Chen, X., Fan, H., Girshick, R., He, K.: Improved baselines with momentum contrastive learning. arXiv:2003.04297 (2020)

8. Fang, T., Lu, N., Niu, G., Sugiyama, M.: Rethinking importance weighting for deep learning under distribution shift. In: Neural Information Processing Systems (2020)
9. Felzenszwalb, P., McAllester, D., Ramanan, D.: A discriminatively trained, multiscale, deformable part model. In: 2008 IEEE conference on Computer Vision and Pattern Recognition. pp. 1–8. IEEE (2008)
10. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of computer and system sciences **55**(1), 119–139 (1997)
11. Friedman, J.H.: Greedy function approximation: A gradient boosting machine. Annals of Statistics **29**, 1189–1232 (2001)
12. Ghosh, A., Kumar, H., Sastry, P.S.: Robust loss functions under label noise for deep neural networks. Proceedings of the AAAI Conference on Artificial Intelligence **31**(1) (2017)
13. Ghosh, A., Lan, A.: Contrastive learning improves model robustness under label noise. arXiv:2104.08984 [cs.LG] (2021)
14. Gretton, A., Smola, A., Huang, J., Schmittfull, M., Borgwardt, K., Schölkopf, B.: Covariate shift by kernel mean matching. Dataset shift in machine learning **3**(4), 5 (2009)
15. Grill, J.B., Strub, F., Altché, F., Tallec, C., Richemond, P., Buchatskaya, E., Doersch, C., Avila Pires, B., Guo, Z., Gheshlaghi Azar, M., Piot, B., kavukcuoglu, k., Munos, R., Valko, M.: Bootstrap your own latent - a new approach to self-supervised learning. In: Advances in Neural Information Processing Systems. vol. 33, pp. 21271–21284 (2020)
16. Gui, X.J., Wang, W., Tian, Z.H.: Towards understanding deep learning from noisy labels with small-loss criterion. In: International Joint Conference on Artificial Intelligence (2021)
17. Han, B., Yao, Q., Yu, X., Niu, G., Xu, M., Hu, W., Tsang, I., Sugiyama, M.: Co-teaching: Robust training of deep neural networks with extremely noisy labels p. 11 (2018)
18. He, K., Fan, H., Wu, Y., Xie, S., Girshick, R.: Momentum contrast for unsupervised visual representation learning. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (June 2020)
19. Hendrycks, D., Mazeika, M., Wilson, D., Gimpel, K.: Using trusted data to train deep networks on labels corrupted by severe noise. In: Advances in Neural Information Processing Systems. vol. 31, pp. 10456–10465 (2018)
20. Huang, J., Gretton, A., Borgwardt, K., Schölkopf, B., Smola, A.J.: Correcting sample selection bias by unlabeled data. In: Advances in neural information processing systems. pp. 601–608 (2007)
21. Jaiswal, A., Babu, A.R., Zadeh, M.Z., Banerjee, D., Makedon, F.: A survey on contrastive self-supervised learning. Technologies **9**(1), 2 (2021)
22. Jiang, L., Meng, D., Zhao, Q., Shan, S., Hauptmann, A.: Self-paced curriculum learning. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 29 (2015)
23. Jiang, L., Zhou, Z., Leung, T., Li, L.J., Fei-Fei, L.: MentorNet: Learning data-driven curriculum for very deep neural networks on corrupted labels. In: International Conference on Machine Learning. vol. 80, pp. 2304–2313 (2018)
24. Jing, L., Tian, Y.: Self-supervised visual feature learning with deep neural networks: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)

25. Kobyzev, I., Prince, S., Brubaker, M.: Normalizing flows: An introduction and review of current methods. IEEE Transactions on Pattern Analysis and Machine Intelligence (2020)
26. Kramer, M.A.: Nonlinear principal component analysis using autoassociative neural networks. AIChE Journal **37**(2), 233–243 (Feb 1991)
27. Kumar, M., Packer, B., Koller, D.: Self-paced learning for latent variable models. In: Advances in Neural Information Processing Systems. vol. 23 (2010)
28. Lee, J., Chung, S.Y.: Robust training with ensemble consensus. In: International Conference on Learning Representations (2020)
29. Li, J., Socher, R., Hoi, S.C.: Dividemix: Learning with noisy labels as semi-supervised learning. In: International Conference on Learning Representations (2019)
30. Li, M., Soltanolkotabi, M., Oymak, S.: Gradient descent with early stopping is provably robust to label noise for overparameterized neural networks. In: International Conference on Artificial Intelligence and Statistics. pp. 4313–4324 (2020)
31. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision. pp. 2980–2988 (2017)
32. Liu, T., Tao, D.: Classification with noisy labels by importance reweighting. IEEE Transactions on Pattern Analysis and Machine Intelligence **38**(3), 447–461 (Mar 2016)
33. Loshchilov, I., Hutter, F.: Sgdr: Stochastic gradient descent with warm restarts. arXiv preprint arXiv:1608.03983 (2016)
34. Maennel, H., Alabdulmohsin, I.M., Tolstikhin, I.O., Baldock, R.J.N., Bousquet, O., Gelly, S., Keysers, D.: What do neural networks learn when trained with random labels? In: Neural Information Processing Systems (2020)
35. Nikodym, O.: Sur une généralisation des intégrales de m. j. radon. Fundamenta Mathematicae **15**(1), 131–179 (1930)
36. Nodet, P., Lemaire, V., Bondu, A., Cornuéjols, A.: Importance reweighting for biquality learning. In: Proceedings of the International Joint Conference on Neural Networks (IJCNN) (2021)
37. Patrini, G., Rozza, A., Menon, A., Nock, R., Qu, L.: Making deep neural networks robust to label noise: a loss correction approach. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2017)
38. Ren, M., Zeng, W., Yang, B., Urtasun, R.: Learning to reweight examples for robust deep learning. In: International Conference on Machine Learning. vol. 80, pp. 4334–4343 (2018)
39. van Rooyen, B., Menon, A., Williamson, R.C.: Learning with symmetric label noise: The importance of being unhinged. In: Neural Information Processing Systems, pp. 10–18 (2015)
40. Shi, Y., Sha, F.: Information-theoretical learning of discriminative clusters for unsupervised domain adaptation. In: ICML (2012)
41. Shu, J., Xie, Q., Yi, L., Zhao, Q., Zhou, S., Xu, Z., Meng, D.: Meta-weight-net: Learning an explicit mapping for sample weighting. In: Neural Information Processing Systems. vol. 32 (2019)
42. Shu, J., Zhao, Q., Xu, Z., Meng, D.: Meta transition adaptation for robust deep learning with noisy labels. arXiv:2006.05697 (2020)
43. Song, H., Kim, M., Park, D., Shin, Y., Lee, J.G.: Learning from noisy labels with deep neural networks: A survey. arXiv:2007.08199 [cs.LG] (2021)
44. Sugiyama, M., Suzuki, T., Kanamori, T.: Density ratio estimation: A comprehensive review (statistical experiment and its related topics) (2010)

45. Wang, Y., Ma, X., Chen, Z., Luo, Y., Yi, J., Bailey, J.: Symmetric cross entropy for robust learning with noisy labels. In: IEEE/CVF International Conference on Computer Vision. pp. 322–330 (2019)
46. Wang, Y., Huang, R., Huang, G., Song, S., Wu, C.: Collaborative learning with corrupted labels. Neural Networks **125**, 205–213 (2020)
47. Xia, X., Liu, T., Wang, N., Han, B., Gong, C., Niu, G., Sugiyama, M.: Are anchor points really indispensable in label-noise learning? In: NeurIPS (2019)
48. Yarowsky, D.: Unsupervised word sense disambiguation rivaling supervised methods. In: 33rd annual meeting of the association for computational linguistics. pp. 189–196 (1995)
49. Zhang, C., Bengio, S., Hardt, M., Recht, B., Vinyals, O.: Understanding deep learning (still) requires rethinking generalization. Communications of the ACM **64**(3), 107–115 (2021)
50. Zhang, H., Yao, Q.: Decoupling representation and classifier for noisy label learning. arXiv:2011.08145 (2020)
51. Zhang, Z., Sabuncu, M.: Generalized cross entropy loss for training deep neural networks with noisy labels. In: Neural Information Processing Systems. vol. 31 (2018)
52. Zheltonozhskii, E., Baskin, C., Mendelson, A., Bronstein, A.M., Litany, O.: Contrast to divide: Self-supervised pre-training for learning with noisy labels. arXiv:2103.13646 [cs.LG] (2021)
53. Zheng, G., Awadallah, A.H., Dumais, S.: Meta label correction for noisy label learning. Proceedings of the AAAI Conference on Artificial Intelligence **35** (2021)

Combining Gaussian Processes with Neural Networks for Active Learning in Optimization

Jiří Růžička[1], Jan Koza[1], Jiří Tumpach[2],
Zbyněk Pitra[1], and Martin Holeňa[1,2,3]

[1] Czech Technical University, Prague, Czech Republic
[2] Charles University, Prague, Czech Republic
[3] Institute of Computer Science, Czech Academy of Sciences, Prague, Czech Republic

Abstract. One area where active learning plays an important role is black-box optimization of objective functions with expensive evaluations. To deal with such evaluations, continuous black-box optimization has adopted an approach called surrogate modelling or metamodelling, which consists in replacing the true black-box objective in some of its evaluations with a suitable regression model, the selection of evaluations for replacement being an active learning task. This paper concerns surrogate modelling in the context of a surrogate-assisted variant of the continuous black-box optimizer Covariance Matrix Adaptation Evolution Strategy. It reports the experimental investigation of surrogate models combining artificial neural networks with Gaussian processes, for which it considers six different covariance functions. The experiments were performed on the set of 24 noiseless benchmark functions of the platform Comparing Continuous Optimizers COCO with 5 different dimensionalities. Their results revealed that the most suitable covariance function for this combined kind of surrogate models is the rational quadratic followed by the Matérn $\frac{5}{2}$ and squared exponential. Moreover, the rational quadratic and squared exponential covariances were found interchangeable in the sense that for no function, no group of functions, no dimension and combination of them, the performance of the respective surrogate models was significantly different.

Keywords: active learning, black-box optimization, artificial neural networks, Gaussian processes, covariance functions

1 Introduction

One area where active learning plays a very important role is *black-box optimization*, in particular optimization of black-box objective functions with expensive evaluations. It is immaterial whether that expensiveness is due to time-consuming computation like in long simulations [15], or due to evaluation in costly experiments like in some areas of science [3]. To deal with such expensive evaluations, continuous black-box optimization has in the late 1990s and early 2000s adopted an approach called *surrogate modelling* or *metamodelling* [6, 12, 14, 32, 40, 43, 46]. In this case, the goal of the surrogate model is to decrease the total number of evaluations of the true objective function. Basically, a surrogate model is any regression model that with a sufficient fidelity, approximates the true black-box objective function and replaces it in some of its evaluations. And the decision in which points to evaluate the expensive black-box objective function, and in which to use its surrogate approximation is an active learning task.

This work-in-progress paper concerns surrogate modelling in the context of a state-of-the-art method for continuous black-box optimization, the Covariance Matrix Adaptation Evolution Strategy (*CMA-ES*) [20, 23]. It reports the first results of our investigation of surrogate models based on combining artificial neural networks (ANNs) with Gaussian processes (GPs). This investigation has been motivated by the importance of surrogate models based on ANNs alone [19,27–29,40,50] and especially on GPs alone [5,6,12–14,31,32,35,47,48], as well as by the high popularity of ANNs in the last 10–15 years. To our knowledge, this is the first time that ANN+GP combinations have been investigated for possible application in surrogate modelling. On the other hand, research into combining parametric ANN models with nonparametric GP models has been around for nearly a decade, at first due to the increasing popularity of neural networks, later also due to recent theoretical results concerning relationships of asymptotic properties of important kinds of ANNs to properties of GPs [33, 37, 39]. The integration of GP with neural learning has been proposed on two different levels:

(i) *Proper integration* of an ANN with a GP, in which the GP forms the final, output layer of the ANN [8, 49].

(ii) Only a *transfer of the layered structure*, which is a crucial feature of ANNs, to the GP context, leading to the concept of deep GPs (DGPs) [7,11,24,25].

In the reported investigation, we employed proper integration, using a GP as the final layer of an ANN.

The rest of the paper is organized as follows. In the next section, the theoretical fundamentals of GP regression and integration with ANNs are recalled. Section 3 describes active learning in a surrogate-assisted variant of CMA-ES. Replacing GPs in that variant with several ANN+GP combinations is then experimentally tested in Section 4. Finally, the concluding Section 5 also outlines our future research plans.

2 Gaussian Processes and Their Integration with Neural Networks

2.1 Gaussian Processes

A *Gaussian process* on a set $\mathcal{X} \subset \mathbb{R}^d, d \in \mathbb{N}$ is a collection of random variables $(f(x))_{x\in\mathcal{X}}$, any finite number of which has a joint Gaussian distribution [45]. It is completely specified by a *mean function* $m_{\mathrm{GP}} : \mathcal{X} \to \mathbb{R}$, typically assumed constant, and by a *covariance function* $\kappa : \mathcal{X} \times \mathcal{X} \to \mathbb{R}$ such that for $x, x' \in \mathcal{X}$,

$$\mathbb{E}f(x) = m_{\mathrm{GP}} \tag{1}$$

$$\operatorname{cov}(f(x), f(x')) = \kappa(x, x'). \tag{2}$$

Therefore, a GP is usually denoted $\mathcal{GP}(m_{\mathrm{GP}}, \kappa)$ or $\mathcal{GP}(m_{\mathrm{GP}}, \kappa(x, x'))$.

The value of $f(x)$ is typically accessible only as a *noisy observation* $y = f(x) + \varepsilon$, where ε is a zero-mean Gausssian noise with a variance $\sigma_n > 0$. Then

$$\mathrm{cov}(y, y') = \kappa(x, x') + \sigma_n^2 \mathbb{I}(x = x'), \tag{3}$$

where $\mathbb{I}$(proposition) equals for a true proposition 1, for a false proposition 0.

Consider now the prediction of the random variable $f(x_\star)$ in a point $x_\star \in \mathcal{X}$ if we already know the observations $y_1, \dots, y_n$ in points $x_1, \dots, x_n$. Introduce the vectors $x = (x_1, \dots, x_n)^\top$, $y = (y_1, \dots, y_n)^\top = (f(x_1) + \varepsilon, \dots f(x_n) + \varepsilon)^\top$, $k_\star = (\kappa(x_1, x_\star), \dots, \kappa(x_n, x_\star))^\top$ and the matrix $K \in \mathbb{R}^{n \times n}$ such that $(K)_{i,j} = \kappa(x_i, x_j) + \sigma_n^2 \mathbb{I}(i = j)$. Then the probability density of the vector y of observations is

$$p(y; m_{\mathrm{GP}}, \kappa, \sigma_n^2) = \frac{\exp\left(-\frac{1}{2}(y - m_{\mathrm{GP}})^\top K^{-1}(y - m_{\mathrm{GP}})\right)}{\sqrt{2\pi \det(K)}}, \tag{4}$$

where $\det(A)$ denotes the determinant of a matrix A. Further, as a consequence of the assumption of Gaussian joint distribution, also the conditional distribution of $f(x_\star)$ conditioned on y is Gaussian, namely

$$\mathcal{N}\left(m_{\mathrm{GP}}(x_\star) + k_\star K^{-1}(y - m_{\mathrm{GP}}), \kappa(x_\star, x_\star) - k_\star^\top K^{-1} k_\star\right). \tag{5}$$

According to (3), the relationship between the observations y and y' is determined by the covariance function κ. In the reported research, we have considered 6 kinds of covariance functions, listed below. In their definitions, the notation $r = \|x' - x\|$ is used, and among the parameters of κ, aka hyperparameters of the GP, frequently encountered are $\sigma_f^2, \ell > 0$, called *signal variance* and *characteristic length scale*, respectively. Other parameters will be introduced for each covariance function separately.

(i) *Linear*: $\kappa_{\mathrm{LIN}}(x, x') = \sigma_0^2 + x^\top x'$, with a bias σ_0^2.

(ii) *Quadratic* is the square of the linear covariance: $\kappa_{\mathrm{QUAD}}(x, x') = (\sigma_0^2 + x^\top x')^2$.

(iii) *Rational quadratic*: $\kappa_{\mathrm{RQ}}(x, x') = \sigma_f^2 \left(1 + \frac{r^2}{2\alpha\ell^2}\right)^{-\alpha}$, with $\alpha > 0$.

(iv) *Squared exponential*: $\kappa_{\mathrm{SE}}(x, x') = \sigma_f^2 \exp\left(-\frac{r^2}{2\ell^2}\right)$.

(v) *Matérn* $\frac{5}{2}$: $\kappa_{\mathrm{MA5}}(x, x') = \sigma_f^2 \left(1 + \frac{\sqrt{5}r}{\ell} + \frac{5r^2}{3\ell^2}\right) \exp\left(-\frac{\sqrt{5}r}{\ell}\right)$.

(vi) One *composite covariance function*, namely the sum of κ_{SE} and κ_{QUAD}: $\kappa_{\mathrm{SE+Q}}(x, x') = \kappa_{\mathrm{SE}}(x, x') + \kappa_{\mathrm{QUAD}}(x, x')$.

2.2 GP as the Output Layer of a Neural Network

An approach integrating a GP into an ANN as its output layer has been independently proposed in [8] and [49]. It relies on the following two assumptions:

1. If n_I denotes the number of the ANN input neurons, then the ANN computes a *mapping* net *of n_I-dimensional input values into the set $\mathcal{X}$* on

which is the GP defined. Consequently, the number n_O of neurons in the last hidden layer fulfills $\mathcal{X} \subset \mathbb{R}^{n_O}$, and the ANN maps an input v into a point $x = \mathrm{net}(v) \in \mathcal{X}$, corresponding to an observation $f(x) + \varepsilon$ governed by the GP (Figure 1). From the point of view of the ANN inputs, the GP is now $\mathcal{GP}(m_{\mathrm{GP}}(\mathrm{net}(v)), \kappa(\mathrm{net}(v), \mathrm{net}(v')))$.

2. The GP mean m_{GP} *is assumed to be a known constant*, thus not contributing to the GP hyperparameters and independent of net.

Due to the assumption 2., the GP depends only on the parameters θ^κ of the covariance function. As to the ANN, it depends on the one hand on the vector θ^W of its weights and biases, on the other hand on the network architecture, which we will treat as fixed before network training.

Consider now n inputs to the neural network, $v_1, \ldots, v_n$, mapped to the inputs $x_1 = \mathrm{net}(v_1), \ldots, x_n = \mathrm{net}(v_n)$ of the GP, corresponding to the observations $y = (y_1, \ldots, y_n)^\top$. Then the log-likelihood of $\theta = (\theta^\kappa, \theta^W)$ is

$$\begin{aligned}\mathcal{L}(\theta) &= \ln p(y; m_{\mathrm{GP}}, \kappa, \sigma_n^2) = \\ &= -\frac{1}{2}(y - m_{\mathrm{GP}})^\top K^{-1}(y - m_{\mathrm{GP}}) \\ &- \ln(2\pi) - \frac{1}{2}\ln\det(K + \sigma_n^2 I_n), \end{aligned} \tag{6}$$

where m_{GP} is the constant assumed in 2., and

$$(K)_{i,j} = \kappa(\mathrm{net}(v_i), \mathrm{net}(v_j)). \tag{7}$$

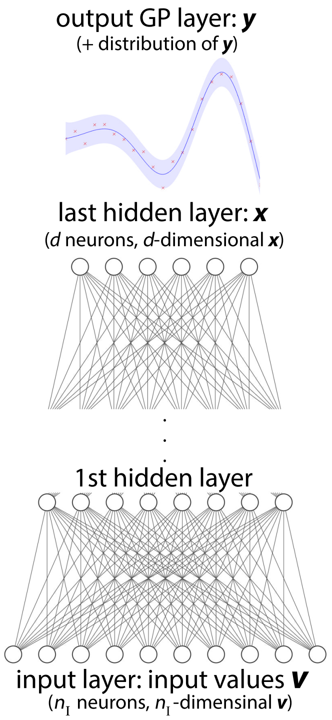

output GP layer: y
(+ distribution of y)
last hidden layer: x
(d neurons, d-dimensional x)
1st hidden layer
input layer: input values V
(n1 neurons, n1-dimensinal v)

Fig. 1. Schema of the integration of a GP into an ANN as its output layer.

Let model training, searching for the vector $(\theta^\kappa, \theta^W)$, be performed in the most simple but, in the context of neural networks, also the most frequent way – as gradient descent. The partial derivatives forming $\nabla_{(\theta^\kappa, \theta^W)}\mathcal{L}$ can be computed as:

$$\frac{\partial \mathcal{L}}{\partial \theta_\ell^\kappa} = \sum_{i,j=1}^{n} \frac{\partial \mathcal{L}}{\partial K_{i,j}} \frac{\partial K_{i,j}}{\partial \theta_\ell^\kappa}, \tag{8}$$

$$\frac{\partial \mathcal{L}}{\partial \theta_\ell^W} = \sum_{i,j,k=1}^{n} \frac{\partial \mathcal{L}}{\partial K_{i,j}} \frac{\partial K_{i,j}}{\partial x_k} \frac{\partial\, \mathrm{net}(v_k)}{\partial \theta_\ell^W}. \tag{9}$$

In (8), the partial derivatives $\frac{\partial \mathcal{L}}{\partial K_{i,j}}$, $i, j = 1, \ldots, n$, are components of the matrix derivative $\frac{\partial \mathcal{L}}{\partial K}$, for which the calculations of matrix differential calculus [36] together with (4) and (6) yield

$$\frac{\partial \mathcal{L}}{\partial K} = \frac{1}{2}\left(K^{-1}yy^\top K^{-1} - K^{-1}\right). \tag{10}$$

3 Surrogate Modelling in the CMA-ES Context

3.1 Surrogate Models for Continuous Black-Box Optimization

Basically, the purpose of surrogate modelling – to approximate an unknown functional dependence – coincides with the purpose of *response surface modelling* in the design of experiments [26, 38]. Therefore, it is not surprising that typical response surface models, i.e., *low order polynomials*, belong also to the most traditional and most successful surrogate models [1, 2, 21, 30, 43]. Other frequently used kinds of surrogate models are *artificial neural networks* of the kind multilayer perceptron (MLP) or radial basis function network [19, 27–29, 40, 50], and the models to which the previous section was devoted – GPs, in surrogate modelling also known as *kriging* [5,6,12–14,31,32,35,47,48]. Occasinally encountered are *support vector regression* [9, 34] and *random forests* [4, 41].

From the point of view of active learning, the most attractive kind of surrogate models are GPs, due to the fact that a GP estimate $f(x)$ of the value of a true objective function for an input x is not a point, but a random variable. Its Gaussian distribution allows to define alternative criteria according to which individuals for evaluation by the true objective function can be selected, most importantly:

- *Probability of improvement* of the estimate $f(x)$ with respect to a reference value V (typically the minimal so far found value of the true objective function),

$$\mathrm{PoI}(f(x); V) = P(f(x) \leq V), \tag{11}$$

  which can be estimated using the Gaussian distribution of the GP.
- *Expected improvement* with respect to V,

$$\mathrm{EI}(f(x), V) = \mathbb{E}(V - f(x))\mathbb{I}(f(x) < V)., \tag{12}$$

3.2 Covariance Matrix Adaptation Evolution Strategy and Its Surrogate-Assisted Variant DTS-CMA-ES

The CMA-ES algorithm performs unconstrained optimization on $\mathbb{R}^d$, by means of iterative sampling of populations sized λ from a d-dimensional Gaussian distribution $\mathcal{N}(m, \sigma^2 C)$, and uses a given parent number μ among the sampled points corresponding to the lowest objective function values, to update the parameters of that distribution. Hence, it updates the expected value m, which is used as the current point estimate of the function optimum, the matrix C and the step-size σ. The CMA-ES is invariant with respect to monotonous transformations of the objective function. Hence, to make use of the evaluations of the objective function in a set of points, it needs to know only the ordering of those evaluations. Details of the algorithm can be found in [20, 23].

During the more than 20 years of CMA-ES existence, a number of surrogate-assisted variants of this algorithm have been proposed, a survey can be found in [5,42]. Here, we will pay attention only to the most recent GP-based among them, the Doubly Trained Surrogate CMA-ES (DTS-CMA-ES) [5], a surrogate-assisted variant of CMA-ES. It employs two GPs f_1 and f_2, trained consecutively, to find an evaluation of the population $x_1, \ldots, x_\lambda$, with f_1 used for active learning of training data for f_2. Due to the CMA-ES invariance with respect to monotonous transformations, evaluates the difference between predictions only according to the difference in the ordering of those predictions, more precisely, according to the ranking difference error (RDE). The RDE of $y \in \mathbb{R}^\lambda$ with respect to $y' \in \mathbb{R}^\lambda$ considering k best components is defined:

$$\underset{\leq k}{\mathrm{RDE}}(y, y') = \frac{\sum_{i,(\rho(y'))_i \leq k} |(\rho(y'))_i - (\rho(y))_i|}{\max_{\pi \in \Pi(\lambda)} \sum_{i=1}^{k} |i - \pi^{-1}(i)|}, \tag{13}$$

where $\Pi(\lambda)$ denotes the set of permutaions of $\{1, \ldots, \lambda\}$ and $\rho(y)$ denotes the ordering of the components of y, i.e., $(\forall y \in \mathbb{R}^\lambda)\ \rho(y) \in \Pi(\lambda)\ \&\ (\rho(y))_i < (\rho(y))_j \Rightarrow y_i \leq y_j$.

The algorithm DTS-CMA-ES is described in Algorithm 1, using the following notation:

- $\mathcal{A}$ for an *archive* – a set of points that have already been evaluated by the true black-box objective function BB;
- $d_{\sigma^2 C}$ for the Mahalanobis distance given by $\sigma^2 C$:

$$d_{\sigma^2 C}(x, x') = \sqrt{(x - x')^\top \sigma^{-2} C^{-1} (x - x')}; \tag{14}$$

- $N_k(x; \mathcal{A})$ for the set of a given number k of $d_{\sigma^2 C}$-nearest neighbours of $x \in \mathbb{R}^d$ with respect to the archive $\mathcal{A}$;
- $f_i(x_1, \ldots, x_\lambda) = (f_i(x_1), \ldots, f_i(x_\lambda))$, for $i = 1, 2$;
- $\mathcal{T}_h = \bigcup_{j=1}^{\lambda} \{x \in N_h(x_j; \mathcal{A}) | d_{\sigma^2 C}(x, x_j) < r_{\max}\}$ with $r_{\max} > 0$ for $h = 1, \ldots, |\mathcal{A}|$;
- $k(\mathcal{A}) = \max\{h | |\mathcal{T}_h| \leq N_{\max}\}$, with $N_{\max} \in \mathbb{N}$;
- ρ_{PoI} for decreasing ordering of $f_1(x_1), \ldots, f_1(x_\lambda)$ according to the probability of improvement with respect to the lowest BB value found so far,

$$i < j \Rightarrow \mathrm{PoI}(\rho_{\mathrm{PoI}}(f_1(x_1, \ldots, x_\lambda)))_i; V) \geq \mathrm{PoI}(\rho_{\mathrm{PoI}}(f_1(x_1, \ldots, x_\lambda)))_j; V), \tag{15}$$

where $V = \min_{x \in \mathcal{A}} BB(x)$.

Algorithm 1 Algorithm DTS-CMA-ES

```
Require: $x_1, \ldots, x_\lambda \in \mathbb{R}^d$, $\mu$, $\mathcal{A}$, $\sigma$ and $C$ – step size and matrix from the CMA-ES
    distribution, $N_{\max} \in \mathbb{N}$ such that $N_{\max} \geq \lambda, r_{\max} > 0, \beta, \epsilon_{\min}, \epsilon_{\max}, \alpha_{\min}, \alpha_{\max} \in$
    $(0, 1)$
 1: if this is the 1st call of the algorithm in the current CMA-ES run then
 2:     set $\alpha = \epsilon = 0.05$
 3: else
 4:     take over the returned values of $\alpha, \epsilon$ from its previous call in the run
 5: end if
 6: Train a Gaussian process $f_1$ on $\mathcal{T}_{k(\mathcal{A})}$, estimating $m_{\mathrm{GP}}$, $\sigma_n, \sigma_f, \ell$ through maximiza-
    tion of the likelihood (4)
 7: Evaluate $BB(x_j)$ for $x_j$ such that $(\rho_{\mathrm{PoI}}(f_1(x_1, \ldots, x_\lambda)))_j \leq \lceil \alpha\lambda \rceil$ and not yet $BB$-
    evaluated
 8: Update $\mathcal{A}$ to $\mathcal{A} \cup \{(x_j | (\rho_{\mathrm{PoI}}(f_1(x_1), \ldots, f_1(x_\lambda)))_j \leq \lceil \alpha\lambda \rceil\}$
 9: Train a Gaussian process $f_2$ on $\mathcal{T}_{k(\mathcal{A})}$, estimating $m_{\mathrm{GP}}$, $\sigma_n$, $\sigma_f$, $\ell$ through maxi-
    mization of the likelihood (4)
10: For $x_j$ such that $(\rho_{\mathrm{PoI}}(f_1(x_1, \ldots, x_\lambda)))_j \leq \lceil \alpha\lambda \rceil$, update $f_2(x_j) = BB(x_j)$
11: Update $\epsilon$ to $(1 - \beta)\epsilon + \beta\,\mathrm{RDE}_\mu(f_1(x_1, \ldots, x_\lambda), (f_2(x_1, \ldots, x_\lambda))$ and $\alpha$ to $\alpha_{\min} +$
    $\max(0, \min(1, \frac{\epsilon - \epsilon_{\min}}{\epsilon_{\max} - \epsilon_{\min}}))$
12: Update the value $f_2(x_j)$ to $f_2(x_j) - \min\{f_2(x_{j'}) | (\rho_{\mathrm{PoI}}(f_1(x_1, \ldots, x_\lambda))_{j'} > \lceil \alpha\lambda \rceil\} +$
    $\min\{f_2(x_{j'}) | (\rho_{\mathrm{PoI}}(f_1(x_1, \ldots, x_\lambda))_{j'} \leq \lceil \alpha\lambda \rceil\}$ for $j$ fulfilling $(\rho_{\mathrm{PoI}}(f_1(x_1, \ldots, x_\lambda))_j >$
    $\lceil \alpha\lambda \rceil$
13: Return $f_2(x_1), \ldots, f_2(x_\lambda), \epsilon, \alpha$
```

4 Experiments with ANN+GP Integration in the DTS-CMA-ES

4.1 Experimental Setup

For the experiments, we have used the 24 noiseless benchmark functions available on a platform *Comparing Continuous Optimizers (COCO)* [10, 22]. Those benchmarks form five groups with different properties:

1. separable functions: f_1 sphere, f_2 separable ellipsoid, f_3 separable Rastrigin, f_4 Büche-Rastrigin, f_5 linear slope;
2. moderately ill-conditioned functions: f_6 attractive sector, f_7 step ellipsoid, f_8 Rosenbrock, f_9 rotated Rosenbrock;
3. highly ill-conditioned functions: f_{10} ellipsoid with high conditioning, f_{11} discus, f_{12} bent cigar, f_{13} sharp ridge, f_{14} different powers;
4. multi-modal functions with global structure: f_{15} non-separable Rastrigin, f_{16} Weierstrass, f_{17} Schaffers F7, f_{18} ill-conditioned Schaffers F7, f_{19} composite Griewank-Rosenbrock;
5. multi-modal weakly structured functions: f_{20} Schwefel, f_{21} Gallagher's Gaussian 101-me points, f_{22} Gallagher's Gaussian 21-hi points, f_{23} Katsuura, f_{24} Lunacek bi-Rastrigin.

All benchmark functions were optimized on the closed cube $[-5, 5]^d$, where d is the dimension of the input space, and the initial CMA-ES population was sampled uniformly on $[-4,4]^d$. For each noiseless function, 25 different variants were used, obtained as follows:

- each of the functions is scalable for any dimension $d \geq 2$, we have used the five dimensions 2, 3, 5, 10, 20;
- for each function in each dimension, 5 different instances were used, mutually differing through translations and/or rotations.

Each variant f of each benchmark function was optimized for $250d$ evaluations unless it was terminated earlier due to indicated convergence. To evaluate the success of optimization at its end, we used the approach used in [22], to calculate the proportion of achieved optimization target in the interval $[10^{-8}, 10^2]$. However, instead of calculating such a score from a given number of discrete targets log-uniformly distributed in that interval as in [22], we calculated its continuous counterpart as the ratio of the logarithmic length $\lambda_{\log}$ between the subinterval of $[10^{-8}, 10^2]$ corresponding to the achieved distance f^* to the minimum of f at the end of the optimization and the whole interval $[10^{-8}, 10^2]$,

$$r = \frac{\lambda_{\log}([10^{-8}, 10^2] \cap [f^*, +\infty))}{\lambda_{\log}([10^{-8}, 10^2])} = \frac{\max(0, \min(10, 2 - \log_{10} f^*))}{10}. \tag{16}$$

For all experiments, we used the existing implementation of DTS-CMA-ES at https://github.com/bajeluk/surrogate-cmaes, into which we implemented the ANN+GP surrogate models using the system GPyTorch [17], our implementaion is available at https://github.com/c0zzy/surrogate-networks. As to the tunable parameters of DTS-CMA-ES, we used the same values as [5]. As to the tunable parameters of GPyTorch, we fixed the five listed in Table 1 and utilized the default values of the remaining.

Finally, for the neural networks in the ANN+GP combinations, we used fully connected multilayer perceptrons with one hidden layer of a sufficiently low number of neurons to assure their trainability with comparatively small archives typically available in the DTS-CMA-ES. More precisely, all the ANNs used the fully connected topologies $(d - n_O - n_O)$ with

$$n_O = \begin{cases} 2 & \text{if } d = 2, \\ 3 & \text{if } d = 3, 5, \\ 5 & \text{if } d = 10, 20. \end{cases} \tag{17}$$

4.2 First Results and Their Discussion

The first work-in-progress results presented in this paper, compare ANN+GP combinations with different covariance functions of the GP. Table 2 reports the scores of each such combination for each noiseless benchmark function, averaged over the 25 combinations of 5 instances and 5 dimensions. In Tables 3 and 4, the scores are reported for the above defined groups of benchmark functions, and for the considered dimension, respectively. Hence, the score averaging in Table 3

Table 1. Settings for important GPyTorch parameters. For all its remaining tunable parameters, the default values were used

Optimizer	Adam
Learning rate	0.0005
Iterations	1000
Noise	0.0001
Length scale bounds	0.01, 100

includes in addition all functions of the respective group, whereas in Table 4 the scores are averaged over the 120 instances of the 24 benchmark functions. In all three tables, the ANN+GP combination with the highest average score is in bold.

Table 2. Score of the 5 compared ANN+GP combinations for each of the noiseless COCO benchmark functions, averaged over the 25 combinations of the 5 instances of that function and the 5 considered dimensions. For each function, the ANN+GP combination with the highest average score is in bold

	κ_{LIN}	κ_{QUAD}	κ_{SE}	κ_{MA5}	κ_{RQ}	$\kappa_{\text{SE+Q}}$		κ_{LIN}	κ_{QUAD}	κ_{SE}	κ_{MA5}	κ_{RQ}	$\kappa_{\text{SE+Q}}$
f_1	0.38	0.45	0.54	0.54	**0.56**	0.47	f_{13}	0.13	0.15	0.22	0.25	**0.27**	0.23
f_2	0.02	0.11	0.22	0.20	**0.26**	0.19	f_{14}	0.41	0.45	**0.55**	0.54	0.54	0.47
f_3	0.09	0.09	0.10	0.10	**0.11**	0.09	f_{15}	0.09	0.09	0.09	0.10	**0.10**	0.09
f_4	0.07	0.07	0.08	0.08	0.08	**0.09**	f_{16}	0.14	0.14	0.20	0.12	**0.15**	0.12
f_5	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	**1.00**	f_{17}	0.26	0.30	0.33	**0.34**	0.33	0.32
f_6	0.09	0.10	0.13	**0.17**	0.15	0.16	f_{18}	0.21	0.25	0.27	0.25	**0.28**	0.27
f_7	0.28	0.38	0.51	0.43	**0.54**	0.45	f_{19}	0.19	0.20	0.20	0.20	0.20	**0.21**
f_8	0.15	0.17	0.29	**0.29**	0.28	0.27	f_{20}	0.17	0.17	0.18	0.17	0.17	**0.18**
f_9	0.15	0.15	0.18	0.24	**0.30**	0.26	f_{21}	**0.19**	0.18	0.16	0.18	0.16	0.17
f_{10}	0.02	0.07	0.10	0.07	**0.10**	0.07	f_{22}	0.15	0.11	**0.16**	0.13	**0.16**	0.14
f_{11}	0.05	0.09	0.12	0.10	**0.13**	0.12	f_{23}	0.17	0.16	0.17	**0.17**	0.17	0.16
f_{12}	0.03	0.07	0.08	0.10	**0.13**	0.09	f_{24}	0.07	0.07	0.07	**0.07**	0.07	0.07

From Tables 2–4, we can see that the highest average score has been by far the most frequently achieved by ANN+GP combinations with rational quadratic covariance functions κ_{RQ}: for 14 out of the 24 functions, 3 out of the 5 groups of functions, and 4 out of the 5 considered dimensions. The rational quadratic covariance function shares the first place with the squared exponential covariance κ_{SE} for the function f_{22} Gallagher's Gaussian 21-hi points, as well as for the dimension 10. In addition, for the function f_5 linear slope, ANN+GP combinations achieve with all considered covariance function the highest possible average score 1. Apart from these shared first places, other covariance functions lead to ANN+GP combinations with the highest average score in the following cases:

Table 3. Score of the 5 compared ANN+GP combinations for each group of the noiseless COCO benchmark functions, averaged over the 100 or 125 (dependent on the group) combinations of all instances of the functions belonging to that group and the 5 considered dimensions. For each function, the ANN+GP combination with the highest average score is in bold

	κ_{LIN}	κ_{QUAD}	κ_{SE}	κ_{MA5}	κ_{RQ}	$\kappa_{\text{SE+Q}}$
Separable	0.317	0.348	0.390	0.388	**0.403**	0.370
Moderately ill-conditioned	0.172	0.204	0.281	0.290	**0.323**	0.288
Highly ill-conditioned	0.134	0.170	0.218	0.218	**0.241**	0.197
Multi-modal functions globally structured	0.182	0.200	**0.221**	0.204	0.218	0.206
Multi-modal weakly structured	**0.153**	0.143	0.151	0.149	0.149	0.147

Table 4. Score of the 5 compared ANN+GP combinations for each considered dimension, averaged over the 120 considered instances of the 24 noiseless COCO benchmark functions. For each dimension, the ANN+GP combination with the highest average score is in bold

	κ_{LIN}	κ_{QUAD}	κ_{SE}	κ_{MA5}	κ_{RQ}	$\kappa_{\text{SE+Q}}$
2	0.268	0.322	0.386	0.389	**0.398**	0.365
3	0.23	0.254	0.326	0.307	**0.365**	0.302
5	0.178	0.194	0.225	0.223	**0.244**	0.224
10	0.162	0.166	**0.185**	0.184	**0.185**	0.174
20	0.124	0.131	0.133	**0.138**	0.131	0.135

– the covariance κ_{SE} for the function f_{14} different powers as well as for the group of multi-modal globally structured functions;
– the Matérn $\frac{5}{2}$ covariance κ_{MA5} for the functions f_6 attractive sector, f_8 Rosenbrock, f_{17} Schaffers F7, f_{23} Katsuura, and f_{24} Lunacek bi-Rastrigin, as well as for the dimension 20;
– the composite covariance $\kappa_{\text{SE+Q}}$ for the functions f_4 Büche-Rastrigin, f_{19} composite Griewank-Rosenbrock, and f_{20} Schwefel;
– the linear covariance κ_{LIN} for the function f_{21} Gallagher's Gaussian 101-me points, as well as for the group of multi-modal weakly structured functions, to which also f_{21} belongs.

The results in Tables 2–4 reflect both systematic differences due to different covariance functions and random differences due to noise. To assess the influence of different covariance functions without the interference of noise, the obtained differences were tested for statistical significance. To this end, the null hypotheses that the means of the random variables that produced the scores for the individual ANN+GP combinations are all identical were tested, using the Friedman's test with post-hoc identification of the pairs that lead to the rejection of the null hypothesis if it is rejected. Both the Friedman test and its post-hoc tests were performed on the family-wise significance level for multiple-hypotheses testing 5%, and the family-wise significances were assessed from the achieved significances of the individual tests (p-values) by means of the Holm method [16].

For individual functions, the Friedman test was always based on the 25 combinations of their 5 instances and the 5 considered dimensions. The test rejected the null hypothesis for all functions except the f_5 linear slope. The result of the post-hoc tests are summarized in Table 5. The value in each row r and column $c \neq r$ tells for how many among the remaining 23 functions the covariance function in the row was as part of an ANN+GP combination significantly better than the one in the column, or equivalently the one column was significantly worse than the one in the row. This means that the ANN+GP combination with the covariance function in the row yielded a higher average score than with the one in the column, and the post-hoc test rejected on the family-wise significance level 5% the hypothesis of equal mean values of the random variables that produced both scores. For individual function groups, the Friedman tests were based on the 100 or 125 combinations of the functions belonging to the group, their 5 instances and the 5 considered dimensions. For individual dimensions, they were based on the 120 combinations of the 24 functions and their dimensions. Both the 5 tests for the function groups and the 5 tests for the dimensions rejected their null hypotheses. The results of their post-hoc tests are summarized in Table 6. In Tables 5 and 6, the value in the cell is in bold if the covariance function in the row was more often significantly better than significantly worse than the one in the column.

Table 5. The number of functions among those 23 for which the null hypothesis of the Friedman test was rejected, for which the covariance function in the row was as part of an ANN+GP combination significantly better than the one in the column. That value is in bold if it is in addition higher than the number of functions for which the covariance function in the row was significantly worse than the one in the column

Covariance function	κ_{LIN}	κ_{QUAD}	κ_{SE}	κ_{MA5}	κ_{RQ}	$\kappa_{\text{SE+Q}}$
κ_{LIN}	–	0	0	0	0	0
κ_{QUAD}	0	–	0	0	0	0
κ_{SE}	**2**	0	–	0	0	**14**
κ_{MA5}	**1**	0	0	–	9	**14**
κ_{RQ}	**6**	**1**	0	**14**	–	**18**
$\kappa_{\text{SE+Q}}$	0	0	9	9	5	–

From Tables 5 and 6, we can observe that the covariance function κ_{LIN} was never either significantly better or significantly worse than κ_{QUAD}. This can be interpreted as interchangeability of both functions from the point of view of being used as covariances in the ANN+GP surrogate models for DTS-CMA-ES. The same relationship holds also for the pairs of covariance functions $(\kappa_{\text{SE}}, \kappa_{\text{MA5}})$ and $(\kappa_{\text{SE}}, \kappa_{\text{RQ}})$. It is particularly important in connection with the last mentioned pair, in view of the fact that κ_{RQ} was the covariance most often yielding the highest score, and κ_{SE} was in this respect the 3rd among the individual bench-

Table 6. The number of function groups (left), respectively considered dimensions (right) for which the covariance function in the row was as part of an ANN+GP combination significantly better than the one in the column. That value is in bold if it is in addition higher than the number of function groups, respectively dimensions for which the covariance function in the row was significantly worse than the one in the column

Covariance function	for function groups						for dimensions					
	κ_{LIN}	κ_{QUAD}	κ_{SE}	κ_{MA5}	κ_{RQ}	$\kappa_{\text{SE+Q}}$	κ_{LIN}	κ_{QUAD}	κ_{SE}	κ_{MA5}	κ_{RQ}	$\kappa_{\text{SE+Q}}$
κ_{LIN}	–	0	0	0	0	0	–	0	0	0	0	0
κ_{QUAD}	0	–	0	0	0	0	0	–	0	0	0	0
κ_{SE}	**4**	**2**	–	0	0	**4**	**4**	**2**	–	0	0	**4**
κ_{MA5}	**4**	**2**	0	–	0	**4**	**4**	**2**	0	–	1	**4**
κ_{RQ}	**4**	**2**	0	**5**	–	**5**	**4**	**2**	0	**4**	–	**4**
$\kappa_{\text{SE+Q}}$	**4**	**1**	1	1	0	–	**4**	**1**	1	1	1	–

mark functions and the 2nd among groups of functions and among the considered dimensions. Altogether, these two interchangeable covariance functions yielded the highest score for 15 from the 24 considered benchmarks, for 4 from the 5 benchmark function groups and for 4 from the 5 considered dimensions.

5 Conclusion ad Future Work

This work-in-progress paper presented an experimental investigation of surrogate models combining artificial neural networks with Gaussian processes in the context of a sophisticated surrogate-assisted variant of the black-box optimizer CMA-ES, the DTS-CMA-ES, which consecutively trains two surrogate models, using the first for active learning of training data for the second. In the experiments, a comprehensive comparison of ANN+GP surrogate models with six different covariance functions was performed on the 24 noiseless benchmark functions of the COCO platform [10, 22] in 5 dimensions. The results revealed that the most suitable covariance function for this combined kind of surrogate models is the rational quadratic, followed by the Matérn $\frac{5}{2}$, and squared exponential. Moreover, the rational quadratic and squared exponential were found interchangeable in the sense that for no function, no group of functions, no dimension, and no function-dimension combination, none of these covariance functions was significantly better than the other.

As usually with work in progress, still very much is left for the future. Most importantly, the ANN+GP surrogate models need to be compared with pure GP surrogates. Superficially, that should be no problem because the original implementation of DTS-CMA-ES uses GPs alone. However, the DTS-CMA-ES implementation relying on the system GPML [44], and our implementation of the ANN+GP surrogates relying on the system GPyTorch [17] do not allow a comparison in which the difference between pure GP and ANN+GP surrogates would reflect only the added combination of both kinds of models. According to our experience, that difference is much more due to the incompatibility between GPML and GPyTorch. The GPML does not include any ANN extension, whereas the GPyTorch includes also pure GPs, however, their predictive acuracy is substantially lower than that of their counterparts with the same covariance function that are implemented in the GPML. Hence, to arrive to an unbiased comparison of ANN+GP and pure GP surrogate models will still need a lot of implementation effort.

Further directions of our future research include on the one hand deep GPs, mentioned already in the Introducion, on the other hand the reconsideration of the approach employed in GPyTorch – training the ANN and the GP forming the combined surrogate model together by means of likelihood maximization. Whereas maximum likelihood is indeed the commonly used objective function for GP learning [45], successful and efficient ANN learning algorithms typically rely on other objectives [18]. Therefore, we would also like to investigate a cyclic interleaving of a GP-learning phase with an ANN-learning phase, where the length of the latter will depend on the relationship of its success to the success of the former.

Finally, we intend to perform research into transfer learning of such combined surrogate models: an ANN-GP model with a deep neural network will be trained on data from many optimization runs, and then the model used in a new run of the same optimizer will be obtained through additional learning restricted only to the GP and the last 1-2 layers of the ANN.

Acknowledgment

The research reported in this paper has been supported by the Czech Science Foundation (GAČR) grant 18-18080S. For J. Tumpach, it has been also partially supported by the Charles University student grant 260575. Computational resources were supplied by the project e-INFRA LM2018140 provided within the program Projects of Large Research, Development and Innovations Infrastructures.

References

1. Auger, A., Brockhoff, D., Hansen, N.: Benchmarking the local metamodel cma-es on the noiseless BBOB'2013 test bed. In: GECCO'13. pp. 1225–1232 (2013)
2. Auger, A., Schoenauer, M., Vanhaecke, N.: LS-CMA-ES: A second-order algorithm for covariance matrix adaptation. In: Parallel Problem Solving from Nature - PPSN VIII. pp. 182–191 (2004)
3. Baerns, M., Holeňa, M.: Combinatorial Development of Solid Catalytic Materials. Design of High-Throughput Experiments, Data Analysis, Data Mining. Imperial College Press / World Scientific, London (2009)
4. Bajer, L., Pitra, Z., Holeňa, M.: Benchmarking Gaussian processes and random forests surrogate models on the BBOB noiseless testbed. In: GECCO'15 Companion. pp. 1143–1150 (2015)
5. Bajer, L., Pitra, Z., Repický, J., Holeňa, M.: Gaussian process surrogate models for the CMA evolution strategy. Evolutionary Computation **27**, 665–697 (2019)

6. Booker, A., Dennis, J., Frank, P., Serafini, D., V., T., Trosset, M.: A rigorous framework for optimization by surrogates. Structural and Multidisciplinary Optimization **17**, 1–13 (1999)
7. Bui, T., Hernandez-Lobato, D., Hernandez-Lobato, J., Li, Y., Turner, R.: Deep gaussian processes for regression using approximate expectation propagation. In: ICML. pp. 1472–1481 (2016)
8. Calandra, R., Peters, J., Rasmussen, C., Deisenroth, M.: Manifold gaussian processes for regression. In: IJCNN. pp. 3338–3345 (2016)
9. Clarke, S., Griebsch, J., Simpson, T.: Analysis of support vector regression for approximation of complex engineering analyses. Journal of Mechanical Design **127**, 1077–1087 (2005)
10. The COCO platform (2016), http://coco.gforge.inria.fr
11. Cutajar, K., Bonilla, E., Michiardi, P., Filippone, M.: Random feature expansions for deep gaussian processes. In: ICML. pp. 884–893 (2017)
12. El-Beltagy, M., Nair, P., Keane, A.: Metamodeling techniques for evolutionary optimization of computaitonally expensive problems: Promises and limitations. In: Proceedings of the Genetic and Evolutionary Computation Conference. pp. 196–203. Morgan Kaufmann Publishers (1999)
13. Emmerich, M., Giannakoglou, K., Naujoks, B.: Single- and multi-objective evolutionary optimization assisted by Gaussian random field metamodels,. IEEE Transactions on Evolutionary Computation **10**, 421–439 (2006)
14. Emmerich, M., Giotis, A., Özdemir, M., Bäck, T., Giannakoglou, K.: Metamodel-assisted evolution strategies. In: PPSN VII. pp. 361–370. ACM (2002)
15. Forrester, A., Sobester, A., Keane, A.: Engineering Design via Surrogate Modelling: A Practical Guide. John Wiley and Sons, New York (2008)
16. Garcia, S., Herrera, F.: An extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all pairwise comparisons. Journal of Machine Learning Research **9**, 2677–2694 (2008)
17. Gardner, J.R., Pleiss, G., Bindel, D., Weinberger, K.Q., Wilson, A.G.: Gpytorch: Blackbox matrix-matrix gaussian process inference with gpu acceleration (2019)
18. Goodfellow, I., Bengio, Y., Courville, A.: Deep Learning. MIT Press, Cambridge (2016)
19. Gutmann, H.: A radial basis function method for global optimization. Journal of Global Optimization **19**, 201–227 (2001)
20. Hansen, N.: The CMA evolution strategy: A comparing review. In: Towards a New Evolutionary Computation. pp. 75–102. Springer (2006)
21. Hansen, N.: A global surrogate assisted CMA-ES. In: GECCO'19. pp. 664–672 (2019)
22. Hansen, N., Auger, A., Ros, R., Merseman, O., Tušar, T., Brockhoff, D.: COCO: a platform for comparing continuous optimizers in a black box setting. Optimization Methods and Software **35**, doi:10.1080/10556788.2020.1808977 (2020)
23. Hansen, N., Ostermaier, A.: Completely derandomized self-adaptation in evolution strategies. Evolutionary Computation **9**, 159–195 (2001)
24. Hebbal, A., Brevault, L., Balesdent, M., Talbi, E., Melab, N.: Efficient global optimization using deep gaussian processes. In: IEEE CEC. pp. 1–12, doi 10.1109/CEC40672.2018 (2018)
25. Hernández-Muñoz, G., Villacampa-Calvo, C., Hernández-Lobato, D.: Deep gaussian processes using expectation propagation and monte carlo methods. In: ECML PKDD. pp. 1–17, paper no. 128 (2020)

26. Hosder, S., Watson, L., Grossman, B.: Polynomial response surface approximations for the multidisciplinary design optimization of a high speed civil transport. Optimization and Engineering **2**, 431–452 (2001)
27. Jin, Y., Hüsken, M., Olhofer, M., B., S.: Neural networks for fitness approximation in evolutionary optimization. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation, pp. 281–306. Springer (2005)
28. Jin, Y., Olhofer, M., Sendhoff, B.: Managing approximate models in evolutionary aerodynamic design optimization. In: CEC 2001. pp. 592–599 (2001)
29. Jin, Y., Olhofer, M., Sendhoff, B.: A framework for evolutionary optimization with approximate fitness functions. IEEE Transactions on Evolutionary Computation **6**, 481–494 (2002)
30. Kern, S., Hansen, N., Koumoutsakos, P.: Local metamodels for optimization using evolution strategies. In: PPSN IX. pp. 939–948 (2006)
31. Kruisselbrink, J., Emmerich, M., Deutz, A., Bäck, T.: A robust optimization approach using kriging metamodels for robustness approximation in the CMA-ES. In: IEEE CEC. pp. 1–8 (2010)
32. Leary, S., Bhaskar, A., Keane, A.: A derivative based surrogate model for approximating and optimizing the output of an expensive computer simulation. Journal of Global Optimization **30**, 39–58 (2004)
33. Lee, J., Bahri, Y., Novak, R., Schoenholz, S., Pennington, J., et al.: Deep neural networks as Gaussian processes. In: ICLR. pp. 1–17 (2018)
34. Loshchilov, I., Schoenauer, M., Sebag, M.: Intensive surrogate model exploitation in self-adaptive surrogate-assisted CMA-ES (saACM-ES). In: GECCO'13. pp. 439–446 (2013)
35. Lu, J., Li, B., Jin, Y.: An evolution strategy assisted by an ensemble of local gaussian process models. In: GECCO'13. pp. 447–454 (2013)
36. Magnus, J., Neudecker, H.: Matrix Differential Calculus with Applications in Statistics and Econometrics. John Wiley and Sons, Chichester (2007)
37. Matthews, A., Hron, J., Rowland, M., Turner, R.: Gaussian process behaviour in wide deep neural networks. In: ICLR. pp. 1–15 (2019)
38. Myers, R., Montgomery, D., Anderson-Cook, C.: Response Surface Methodology: Proces and Product Optimization Using Designed Experiments. John Wiley and Sons, Hoboken (2009)
39. Novak, R., Xiao, L., Lee, J., Bahri, Y., Yang, G., et al.: Bayesian deep convolutional networks with many channels are Gaussian processes. In: ICLR. pp. 1–35 (2019)
40. Ong, Y., Nair, P., Keane, A., Wong, K.: Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation. pp. 307–331. Springer (2005)
41. Pitra, Z., Repický, J., Holeňa, M.: Boosted regression forest for the doubly trained surrogate covariance matrix adaptation evolution strategy. In: ITAT 2018. pp. 72–79 (2018)
42. Pitra, Z., Hanuš, M., Koza, J., Tumpach, J., Holena, M.: Interaction between model and its evolution control in surrogate-assisted cma evolution strategy. In: GECCO '21: Genetic and Evolutionary Computation Conference. pp. 528–536 (2021)
43. Rasheed, K., Ni, X., Vattam, S.: Methods for using surrogate modesl to speed up genetic algorithm oprimization: Informed operators and genetic engineering. In: Jin, Y. (ed.) Knowledge Incorporation in Evolutionary Computation. pp. 103–123. Springer (2005)
44. Rasmussen, C., Williams, C.: Gpml 4.0, matlab toolbox for gaussian processes for machine learning, http://www.gaussianprocess.org/gpml/code/matlab/doc/

45. Rasmussen, C., Williams, C.: Gaussian Processes for Machine Learning. MIT Press, Cambridge (2006)
46. Ratle, A.: Kriging as a surrogate fitness landscape in evolutionary optimization. Artificial Intelligence for Engineering Design, Analysis and Manufacturing **15**, 37–49 (2001)
47. Ulmer, H., Streichert, F., Zell, A.: Evolution strategies assisted by Gaussian processes with improved pre-selection criterion. In: IEEE CEC. pp. 692–699 (2003)
48. Volz, V., Rudolph, G., Naujoks, B.: Investigating uncertainty propagation in surrogate-assisted evolutionary algorithms. In: GECCO'17. pp. 881–888 (2017)
49. Wilson, A., Hu, Z., Salakhutdinov, R., Xing, E.: Deep kernel learning. In: ICAIS. pp. 370–378 (2016)
50. Zhou, Z., Ong, Y., Nair, P., Keane, A., Lum, K.: Combining global and local surrogate models to accellerate evolutionary optimization. IEEE Transactions on Systems, Man and Cybernetics. Part C: Applications and Reviews **37**, 66–76 (2007)

Stochastic Adversarial Gradient Embedding for Active Domain Adaptation

Victor Bouvier[1,2], Philippe Very[3], Clément Chastagnol[4], Myriam Tami[1], and Céline Hudelot[1]

[1] Université Paris-Saclay, CentraleSupélec, firstname.name@centralesupelec.fr
[2] Sidetrade, vbouvier@sidetrade.com
[3] Talan, philippe.very@talan.com
[4] IQVIA, clement.chastagnol@iqvia.com

Abstract. Unsupervised Domain Adaptation (UDA) bridges the gap between a labelled source domain and an unlabelled target domain. In this paper, we improve adaptation by guiding the model with actively annotated target data. This problem, named Active Domain Adaptation (ADA) is of practical interest as it is sometimes possible to annotate a small budget of target data in many applications. We introduce **S**tochastic **a**dversarial **g**radient **e**mbedding (Sage), an embedding for estimating the impact of annotating a target sample on adaptation. Sage measures the variation of the transferability loss gradient, before and after annotation. Additionally, we investigate various procedures for incorporating a small subset of labelled target samples when learning domain invariant representations. Our experiments on challenging benchmarks demonstrate that a small effort of active annotation with Sage improves adaptation substantially. Importantly, with a comparable labelling budget, Sage performs better than its semi-supervised counterpart while having more realistic assumptions.

Keywords: Domain Adaptation · Active Learning · Invariant Representations.

1 Introduction

When provided with a large amount of labelled data, deep neural networks have dramatically improved the state-of-the-art in various applications [19]. However, when deployed in the real world, where data may slightly differ from the training data, deep models often fail to generalize out of the training distribution [5]. Nevertheless, deep nets can learn data representations transferable to new tasks or new domains if some labelled data from the new distribution are available [36]. Acquiring a sufficient amount of labelled data is often impossible, and large scale annotation is often cost-prohibitive. In contrast, unlabelled data are much more convenient to obtain. This observation has motivated the field of *Unsupervised Domain Adaptation* (UDA) [23,26] for bridging the gap between a labelled *source domain* and an unlabelled *target domain*. Learning

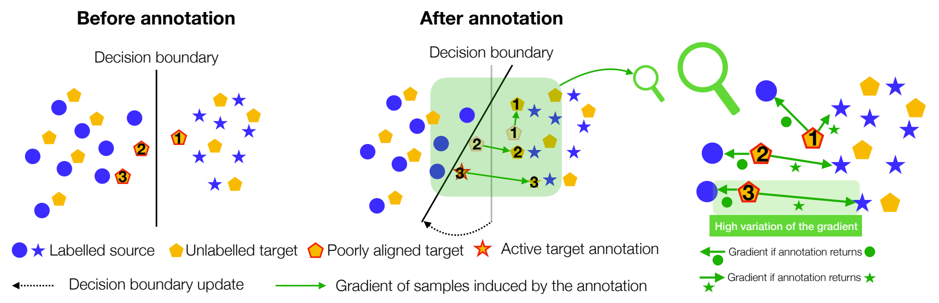


Fig. 1. Effect of annotation of a target sample selected by Sage (*best viewed in colors*). Binary classification problem ($\bullet$ *vs* $\star$) where source samples are blue and target samples are orange. Before annotation, the class-level alignment is not satisfactory leading to a potential negative transfer (poorly aligned target samples tagged as **1**, **2** and **3**). We estimate which sample should be primarily annotated by measuring the variation of the representations' transferability gradient, before and after annotation. We observe the highest variation is obtained for target sample **3**, which is sent to an oracle. The oracle annotation returns class $\star$, validating the suspicion of negative transfer. This leads to an update of the decision boundary, which pushes **1**, **2**, and **3** into class $\star$, resulting in a better class-level alignment of representations.

domain *Invariant Representations* has led to significant progress [13,21,22]. By fooling a discriminator trained to separate the source from the target domain, the feature extractor removes domain-specific information from representations. Thus, a classifier trained from those representations with source labelled data is expected to perform reasonably well in the target domain [6].

However, those methods perform significantly worse than their fully supervised counterparts. To this purpose, Semi-Supervised Domain Adaptation (SSDA) has been studied in [31] through a Mini-Max Entropy objective (MME). Nevertheless, assuming that at least one target labelled sample represents a class, thus involving information about target labels, SSDA is built on assumptions that are unlikely to be met in practice. A more realistic scenario would be to guide adaptation by selecting for annotation a pool target unlabelled instances. This new paradigm referred to as *Active Domain Adaptation* (ADA), is often encountered in real-world applications. To our knowledge, only a few prior works address ADA [9,27,30,34]. In particular, the recent work of Su et al. [34] is the first that uses domain adversarial learning for *Active Learning* (AL).

In this paper, we address ADA reserving the annotation budget for target samples for which their annotations are likely to guide adaptation. In contrast to [34], which selects a diverse set of poorly adapted target samples based on a classical criterion of uncertainty, we estimate the impact of annotation on the representations' transferability. To this purpose, we introduce **S**tochastic **a**dversarial **g**radient **e**mbedding (Sage), an embedding of target samples, whose norm estimates precisely this impact. Our approach also promotes diversity in the annotation. We follow [3] and select target samples for which Sage spans on diverse directions using the `k-means++` initialization [1]. Since access to some labelled data from the target domain brings us back to SSDA, we investigate the role of MME in this context.

We organize the rest of the paper as follows. First, we provide a brief overview of Domain Adversarial Learning for UDA. Importantly, we expose a soft-class conditioning adversarial loss, which reflects the transferability error of domain invariant representations [7]. Second, we present the details of Sage while providing theoretical insights that AL can improve representations' transferability in the third section. Finally, we conduct an empirical study on several benchmarks that support our ADA approach.

2 Background

Notations. Let us consider three random variables; X the input data, Z the representations and Y the labels, defined on spaces $\mathcal{X}$, $\mathcal{Z} \subset \mathbb{R}^m$ where m is the dimension of the representation, and $\mathcal{Y}$ such that $|\mathcal{Y}| = C$ for some integers C, respectively. We note realizations with lower cases, x, z and y, respectively. Those random variables may be sampled from two and different distributions: the *source* distribution p_S *i.e.*, data where the model is trained and the *target* distribution p_T *i.e.*, data where the model is evaluated. Labels are one-hot encoded *i.e.*, $y \in [0,1]^C$ with $\sum_c y_c = 1$ where C is the number of classes. We use the index notation S and T to differentiate source and target quantities. We define the hypothesis class $\mathcal{H}$ as a subset of functions from $\mathcal{X}$ to $\mathcal{Y}$ which are the composition of a representation class Φ (mappings from $\mathcal{X}$ to $\mathcal{Z}$) and a classifier class $\mathcal{F}$ (mappings from $\mathcal{Z}$ to $\mathcal{Y}$) *i.e.*, $h := f\varphi := f \circ \varphi \in \mathcal{H}$ where $f \in \mathcal{F}$ and $\varphi \in \Phi$. For $D \in \{S, T\}$ and a hypothesis $h \in \mathcal{H}$, we introduce the error in domain D, $\varepsilon_D(h) := \mathbb{E}_D[\ell(h(X), Y)]$ where ℓ is the L^2 loss $\ell(y, y') = ||y - y'||^2$ and $h(x)_c$ is the probability of x to belong to class c. We note the source domain data $(x_i^S, y_i^S)_{1 \le i \le n_S}$ and the target domain data $(x_j^T)_{1 \le j \le n_T}$.

Domain Adversarial Learning. The seminal works from [13,21], and their theoretical ground [6], have led to a wide variety of methods based on domain invariant representations [22,20,10]. A representation φ and a classifier f are learned by achieving a trade-off between source classification error and domain invariance of representations by fooling a discriminator trained to separate the source from the target domain:

$$L(\varphi, f) := L_S(\varphi, f) - \lambda \cdot \inf_{d \in \mathcal{D}} L_{\text{INV}}(\varphi, d) \tag{1}$$

where $L_S(\varphi, f) := \mathbb{E}_S[-Y \cdot \log(f\varphi(X))]$ is the cross-entropy loss in the source domain, $L_{\text{INV}}(\varphi, d) := \mathbb{E}_S[\log(1 - d(\varphi(X)))] + \mathbb{E}_T[\log(d(\varphi(X))]$ is the adversarial loss and $\mathcal{D}$ is the set of discriminators *i.e.* mapping from $\mathcal{Z}$ to $[0,1]$. In practice, $\inf_{d \in \mathcal{D}}$ is approximated using a *Gradient Reversal Layer* [13].

Transferability loss for class-level invariance. Promoting class-level domain invariance improves the transferability of representations [22]. Recently, the work [7] introduces the *transferability loss*, noted L_{TSF}, which adds class-conditioning in the adversarial loss by computing a scalar product between labels y and a class-level discriminator d defined as a mapping from $\mathcal{Z}$ to $[0,1]^C$. Since labels are not available in the target domain at train time, predicted labels $\hat{y} := f\varphi(x)$ are used. This approach is referred to as *soft-class conditioning*:

$$L(\varphi, f) := L_S(\varphi, f) - \lambda \cdot \inf_{\mathsf{d}\in\mathsf{D}} L_{\text{TSF}}(\varphi, \hat{y}, \mathsf{d}) \tag{2}$$

where $L_{\text{TSF}}(\varphi, \hat{y}, \mathsf{d}) := \mathbb{E}_S[Y \cdot \log(1 - \mathsf{d}(\varphi(X)))] + \mathbb{E}_T[\hat{Y} \cdot \log(\mathsf{d}(\varphi(X))]$ is the transferability loss and D is the set of class-level discriminators *i.e.*, mappings from $\mathcal{Z}$ to $[0,1]^C$. In this work, we explore the role of active annotation on a small subset of the target domain in order to improve the transferability of representations. Methods based on L_{TSF} as adaptation loss are flagged as TSF.

3 Proposed Method

3.1 Motivations

Gradient-based selection, as shown in Badge [3], is promising in AL. In contrast to Badge, which focuses on the network's predictions, we discuss the role of representations' transferability. To this purpose, we introduce, in the following, the *adversarial gradient* that reflects the lack of transferability of a target sample. From this gradient, we expose a query that efficiently incorporates the domain shift problem in ADA. Let a target sample $x \sim p_T$ with representation $z := \varphi(x) \in \mathbb{R}^m$, we start by describing the effect of annotating the sample x on the gradient descent update of 2. We define the *adversarial gradient* $\mathbf{g}_x$ of x as the gradient of the discriminator loss w.r.t the representation z:

$$\mathbf{g}_x := -\frac{\partial \log(\mathsf{d}(z))}{\partial z} \in \mathbb{R}^{C\times m}, \quad \text{where } \mathsf{d}(z) \in [0,1]^C \tag{3}$$

Following the expression of the transferability loss L_{TSF}, the contribution of a sample x to the gradient update (2), before and after its annotation, is:

$$\underbrace{\left\{\theta \leftarrow \theta - \alpha\frac{\partial z}{\partial \theta} \cdot (\hat{y} \cdot \mathbf{g}_x)\right\}}_{\textit{Before annotation}} \longrightarrow \underbrace{\left\{\theta \leftarrow \theta - \alpha\frac{\partial z}{\partial \theta} \cdot (y \cdot \mathbf{g}_x)\right\}}_{\substack{\textit{After annotation} \\ y\sim\text{Oracle}(x)}}$$

where $\partial z/\partial\theta$ is the jacobian of the representations with respect to the deep network parameters θ *i.e.*, $z := \varphi_\theta(x)$, $\hat{y} := f\varphi_\theta(x)$ is the current label estimation and α is some scaling parameter. Before the annotation, the gradient vector can be written as a weighted sum of $\mathbf{g}_x$ *i.e.*, $\hat{y} \cdot \mathbf{g}_x \in \mathbb{R}^m$, reflecting the class probability of x. Annotating the sample x has the effect of setting, once and for all, a direction of the gradient ($y \cdot \mathbf{g}_x$). Based on this observation, we can measure the annotation procedure's ability to learn more transferable representations by its tendency to change the path of the gradient descent *i.e.*, how $y \cdot \mathbf{g}_x$ may differ with $\hat{y} \cdot \mathbf{g}_x$.

3.2 Positive Orthogonal Projection (POP)

In the rest of the paper, we consider $\mathbf{g}_x \in \mathbb{R}^{C\times m}$ as a stochastic vector of $\mathbb{R}^m$ with realizations lying in $\mathcal{G}_x := \{\mathbf{g}_x^1, ..., \mathbf{g}_x^C\}$ where $\mathbf{g}_x^c = (-\partial \log(\mathsf{d}(z))/\partial z)_c$. When provided the label through an oracle *i.e.*, $y \sim \text{Oracle}(x)$, we obtain $\mathbf{g}_x^y \in \mathcal{G}_x$, a realization of $\mathbf{g}_x$. Before annotation, the direction of the gradient is the *mean* of $\mathbf{g}_x$ where $\mathcal{G}_x$ is provided with the class probability given by the classifier's output $h(x)$. More precisely, the probability of observing $\tilde{\mathbf{g}}_x = \tilde{\mathbf{g}}_x^c$ is $h(x)_c$, then, the *mean* of $\mathbf{g}_x$, noted $\mathbb{E}_h[\mathbf{g}_x]$ is defined as follows:

$$\mathbb{E}_h[\mathbf{g}_x] := \mathbb{E}_{y\sim h(x)}\left[\mathbf{g}_x^y\right] = h(x) \cdot \mathbf{g}_x \in \mathbb{R}^m \tag{4}$$

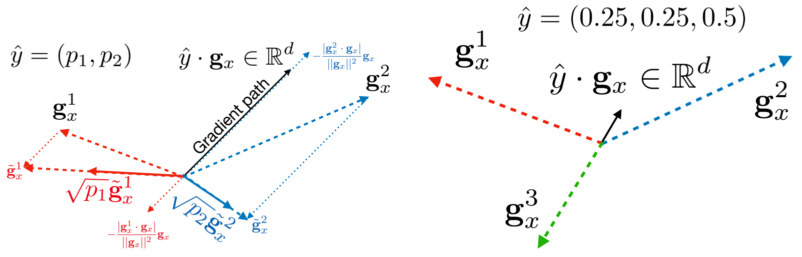
Therefore, the tendency to modify the direction of the gradient is reflected by a high discrepancy between $\mathbb{E}_h[\mathbf{g}_x]$ and $\mathbf{g}_x^y$ for $y \sim \text{Oracle}(x)$. To quantify this discrepancy, we consider variations in both direction and magnitude. To find a good trade-off between these two requirements, we remove the mean direction of the gradient $\mathbb{E}_h[\mathbf{g}_x]$ to $\mathbf{g}_x$ by computing a *Positive Orthogonal Projection* (POP), noting $\lambda := |\mathbf{g}_x \cdot \mathbb{E}_h[\mathbf{g}_x]|/||\mathbb{E}_h[\mathbf{g}_x]||^2$;

$$\tilde{\mathbf{g}}_x := \mathbf{g}_x - \lambda \mathbb{E}_h[\mathbf{g}_x] \tag{5}$$

We motivate the use $|\mathbf{g}_x \cdot \mathbb{E}_h[\mathbf{g}_x]|$ rather than $\mathbf{g}_x \cdot \mathbb{E}_h[\mathbf{g}_x]$ for the standard orthogonal projection. On the one hand, if the annotation provides a gradient with the same direction as the expected gradient *i.e.*, the annotation reinforces the prediction, $\tilde{\mathbf{g}}_x$ is null. On the other hand, if the annotation provides a gradient with an opposite direction to the expected gradient *i.e.*, the annotation contradicts the prediction, the norm of $\tilde{\mathbf{g}}_x$ increases. Therefore, target samples x for which we expect the highest impact on the transferability, are those with the highest norm of $\tilde{\mathbf{g}}_x$. Since λ involves an absolute value, we refer to it as a *positive* orthogonal projection. An illustration is provided in Figure 2. Since $\tilde{\mathbf{g}}_x$ is stochastic, we need additional tools to define a norm operator properly on it.

3.3 Stochastic Adversarial Gradient Embedding

It seems natural to quantify the norm of the stochastic vector $\tilde{\mathbf{g}}_x$ as the square root of the mean of $\tilde{\mathbf{g}}_x$'s norm: $||\tilde{\mathbf{g}}_x||_h := (\mathbb{E}_{y\sim h(x)}\left[||\tilde{\mathbf{g}}_x^y||^2\right])^{1/2}$. However, given x_1 and x_2, how to quantify the discrepancy between $\mathbf{g}_{x_1}$ and $\mathbf{g}_{x_2}$? The difficulty results from the fact that $h(x_1) \neq h(x_2)$ in general. Simply using $\mathbb{E}_{y_1\sim h(x_1), y_2\sim h(x_2)}$ $\left[||\mathbf{g}_{x_1}^{y_1} - \mathbf{g}_{x_2}^{y_2}||^2\right])^{1/2}$ leads to an operator that returns a non-null discrepancy between x and itself if $h(x)$ is not a one-hot vector. To address this issue, we


$\hat{y} = (p_1, p_2)$
$\hat{y} \cdot \mathbf{g}_x \in \mathbb{R}^d$
Gradient path
$\mathbf{g}_x^1$
$\mathbf{g}_x^2$
$\tilde{\mathbf{g}}_x^1$
$\tilde{\mathbf{g}}_x^2$
$\sqrt{p_1}\tilde{\mathbf{g}}_x^1$
$\sqrt{p_2}\tilde{\mathbf{g}}_x^2$
$-\frac{|\mathbf{g}_x^1 \cdot \mathbf{g}_x|}{||\mathbf{g}_x||^2}\mathbf{g}_x$
$-\frac{|\mathbf{g}_x^2 \cdot \mathbf{g}_x|}{||\mathbf{g}_x||^2}\mathbf{g}_x$
$\hat{y} = (0.25, 0.25, 0.5)$
$\hat{y} \cdot \mathbf{g}_x \in \mathbb{R}^d$
$\mathbf{g}_x^1$
$\mathbf{g}_x^2$
$\mathbf{g}_x^3$

(a) Sage (b) Poor local min

Fig. 2. (a) Visualisation of $\mathsf{S}(x) = (\sqrt{p_1}\tilde{\mathbf{g}}_x^1, \sqrt{p_2}\tilde{\mathbf{g}}_x^2)$. Here $\tilde{\mathbf{g}}_x^2 \perp (\hat{y} \cdot \mathbf{g}_x)$ since $\hat{y} \cdot \mathbf{g}_x$ and $\mathbf{g}_x^2$ have a similar direction while $|\tilde{\mathbf{g}}_x^1 \cdot \mathbf{g}_x| \geq |\mathbf{g}_x^1 \cdot \mathbf{g}_x|$ since $\mathbf{g}_x^1$ as a component in the opposite direction of $\hat{y} \cdot \mathbf{g}_x$. (b) Illustration of a case where the transferability loss is close to a local minimum ($\hat{y} \cdot \mathbf{g}_x \approx 0$), but the stochastic gradients ($\mathbf{g}_x^y$ for $y \in \{1, 2, 3\}$) have a high norm. Here, the annotation chooses one of the gradients resulting in a strong update of the model.

suggest to embed x, through a mapping S named *Stochastic adversarial gradient embedding* (Sage):

$$\mathsf{S}(x) := (\sqrt{h(x)_1}\tilde{\mathbf{g}}_x^1, ..., \sqrt{h(x)_C}\tilde{\mathbf{g}}_x^C) \in \mathbb{R}^{C \times m} \tag{6}$$

By choosing $\sqrt{h}$, we guarantee that $||\mathsf{S}(x)|| = ||\tilde{\mathbf{g}}_x||_h$ while offering a proper discrepancy between $\mathbf{g}_{x_1}$ and $\mathbf{g}_{x_2}$ with $||\mathsf{S}(x_1) - \mathsf{S}(x_2)||$. Crucially, both the norm and the distance computed on Sage do not involve the target labels, making it relevant for UDA since target labels are unknown. An illustration of Sage is provided in Figure 2.

3.4 Increasing Diversity of Sage (`k-means++`)

As aforementioned, the higher the norm of $||\mathsf{S}(x)||$, the greater the expected impact of annotating sample x on the transferability of representations. A naive strategy of annotation would be to rank target samples by their Sage norm ($||\mathsf{S}(x)||$). The drawback is to acquire labels for a not IID batch from the target distribution, a problem referred to as the challenge of *diversity* in AL [33]. In certain pathological cases (e.g. the selection of very similar samples or samples of the same class), the IID violation may degrade the performance in the target domain. To label useful target samples (*i.e.*, high $||\mathsf{S}(x)||$) while acquiring a representative batch of the target distribution, we follow [3] by selecting samples with high $||\mathsf{S}(x)||$ which span in various directions. This is performed using the `k-means++` initialization [1]. The procedure `Sage` is detailed in Algorithm 1 for a given budget b of annotation. Importantly, sampling diverse target samples with high impact on transferability results from the construction of an embedding (Sage) suitable with `k-means++`.

Algorithm 1 $\texttt{Sage}(\mathcal{U}_T, b, f, \varphi, \mathsf{d})$: Sage with diversity (`k-means++`)

Input: $\mathcal{U}_T$: Unlabelled target data, budget b, representation φ, classifier f, discriminator d

```
1: Computes $\mathsf{S}(x_u)$ for $x_u \in \mathcal{U}_T$                    ▷ Depends on both $f$ and $\varphi$.
2: $\mathcal{A} \leftarrow \{\text{argmax}_{x_u \in \mathcal{U}_T} ||\mathsf{S}(x_u)||\}$      ▷ Select sample with the highest Sage norm.
3: while $|\mathcal{A}| < b$ do                         ▷ Apply k-means++ on Sage embedding.
4:     $\mathcal{A} \leftarrow \mathcal{A} \cup \{\underset{x_u \in \mathcal{U}_T}{\text{argmax}} \, \underset{x_a \in \mathcal{A}}{\min} ||\mathsf{S}(x_u) - \mathsf{S}(x_a)||\}$
5: end while
6: Return $\mathcal{A}$
```

3.5 Semi-Supervised Domain Adaptation (SSDA)

When acquiring labels in the target domain, we are in the Semi-Supervised Domain Adaptation (SSDA) setting. To this purpose, we note $\mathcal{L}_S$ and $\mathcal{L}_T$ the sets of labelled samples from the source and the target domains, respectively. We study three strategies, referred to as $S{\cup}T$, $S{+}T$ and MME [31]. They incorporate labelled samples into adaptation through an additional loss Ω, called a SSDA regularizer:

$$\Omega_{S\cup T}(f,\varphi) := L_{\mathcal{L}_S \cup \mathcal{L}_T}(f,\varphi) \tag{7}$$

$$\Omega_{S+T}(f,\varphi) := L_{\mathcal{L}_S}(f,\varphi) + L_{\mathcal{L}_T}(f,\varphi) \tag{8}$$

noting $L_{\mathcal{L}}(f,\varphi)$ the empirical cross-entropy of $f\varphi$ computed on some labelled dataset $\mathcal{L}$. Note that Ω_{S+T} gives more importance to target labelled samples compared to $\Omega_{S\cup T}$, especially in the small budget regime (*i.e.*, when the budget b is such that $b \ll |\mathcal{L}_S|$). As a strong baseline exists in SSDA, we design Ω following the minimax entropy (MME) [31]. Noting $H_{\mathcal{U}_T}(h) := -\frac{1}{|\mathcal{U}_T|}\sum_{x\in\mathcal{U}_T} h(x)\cdot \log h(x)$, the entropy of unlabelled samples $\mathcal{U}_T$, the MME objective is:

$$\begin{cases} \Omega_{\text{MME}}(f) := \Omega_{S+T}(f,\varphi) - \lambda H_{\mathcal{U}_T}(f\varphi) \\ \Omega_{\text{MME}}(\varphi) := \Omega_{S+T}(f,\varphi) + \lambda H_{\mathcal{U}_T}(f\varphi) \end{cases} \tag{9}$$

where $f := \sigma\left(\frac{1}{T} W \circ \ell_2\right)$ ($\ell_2(\mathbf{f}) := \mathbf{f}/||\mathbf{f}||_2$ is the L^2 normalization of features and $W \in \mathbb{R}^{C\times m}$ is a linear layer), $\lambda = 0.1$, $T = 0.05$ and σ is the softmax layer.

3.6 Training procedure

The training procedure is described in Algorithm 2. First, we train the model by UDA following the training procedure from [7]. Second, for a given number of iterations, we select by `Sage` (See Algorithm 1) b samples to send to the Oracle. Then, we perform UDA provided with the knowledge of newly labelled samples, that is using a SSDA regularizer Ω combined with soft-class conditioning loss L_{TSF}. We describe the gradient descent step in the following. First, given a loss L, Given a SSDA regularizer Ω (See Section 3.5), the gradient descent step is defined as follows, for some $\alpha > 0$:

$$(f,\varphi,\mathsf{d}) \leftarrow (f,\varphi,\mathsf{d}) - \alpha \nabla_{(f,\varphi,\mathsf{d})}\left(\hat{\Omega}(f,\varphi) + \lambda \hat{L}_{\text{TSF}}(f,\varphi)\right) \tag{10}$$

Algorithm 2 Training procedure

```
Input: Labelled source samples $\mathcal{L}_S$, Unlabelled target samples $\mathcal{U}_T$, budget $b$, annotation rounds $r$, iterations $n_{\text{it}}$, SSDA regularizer $\Omega$:
 1: $\mathcal{L}_T \leftarrow \{\}, \mathcal{U}'_T \leftarrow \mathcal{U}_T$        ▷ Initializes the labelled target samples.
 2: $f, \varphi, \mathsf{d} \leftarrow$ UDA as described in [7]        ▷ Pretraining before Active Learning.
 3: for $b$ rounds of annotations do
 4:     $\mathcal{A} \leftarrow \texttt{Sage}(\mathcal{U}'_T, b, f, \varphi, \mathsf{d})$        ▷ Selects samples for annotation.
 5:     $\mathcal{L} \leftarrow$ Oracle($\mathcal{A}$)        ▷ Sends samples to an Oracle.
 6:     $\mathcal{L}_T \leftarrow \mathcal{L}_T \cup \mathcal{L}$        ▷ Adds newly labelled samples.
 7:     $\mathcal{U}'_T \leftarrow \mathcal{U}'_T \backslash \mathcal{A}$        ▷ Removes newly labelled samples.
 8:     for $n_{\text{it}}$ iterations do
 9:         Sample a source labelled batch $\mathcal{B}^\ell_S$ from $\mathcal{L}_S$
10:         Sample a source labelled batch $\mathcal{B}^\ell_T$ from $\mathcal{L}_T$
11:         Sample a source labelled batch $\mathcal{B}^u_T$ from $\mathcal{U}_T$        ▷ (Not from $\mathcal{U}'_T$).
12:         $f, \varphi, \mathsf{d} \leftarrow$ Gradient descent update from Equation 10.
13:     end for
14: end for
15: Return: $f, \varphi$
```

where for a given loss L, we note its batch-wise computation $\hat{L}$ when provided with batches of source labelled samples $\mathcal{B}^\ell_S$ from $\mathcal{L}_S$, a source labelled samples $\mathcal{B}^\ell_T$ from $\mathcal{L}_T$, a source labelled samples $\mathcal{B}^u_T$ from $\mathcal{U}_T$. Notably, $\mathcal{B}^\ell_S$ and $\mathcal{B}^\ell_T$ are involved for computing $\hat{\Omega}$ (eventually $\mathcal{B}^u_T$ for $\hat{\Omega}_{\text{MME}}$) while $\mathcal{B}^\ell_S$ and $\mathcal{B}^u_T$ are involved for computing $\hat{L}_{\text{TSF}}$.

4 Theoretical Analysis

4.1 General bound

We provide a theoretical analysis of guiding adaptation with AL. It leverages recent results from [7]. Our insight is that some labelled data from the target domain, when combined with source labelled data, are likely to improve the target error. For instance, minimizing Ω_{S+T} may result in a better performing classifier than simply minimizing the source cross-entropy loss $L_{\mathcal{L}_S}$. The theoretical framework from Bouvier et al. [7] allows to quantify precisely how it impacts representations' transferability. Noting $h_S := \text{argmin}_{h\in\mathcal{H}}\varepsilon_S(h)$ and $h_\Omega := \text{argmin}_{h\in\mathcal{H}}\Omega(h)$ such that $\varepsilon_T(h_\Omega) \leq \beta\varepsilon_T(h_S)$ for some $\beta < 1$ *i.e.*, h_Ω improves the target error compared to h_S, the work [7] bounds the target error;

$$\varepsilon_T(h_\Omega) \leq \rho(\varepsilon_S(h_S) + 8\tau + \eta) \quad \text{where} \quad \rho := \frac{\beta}{1-\beta} \tag{11}$$

where $\tau := \sup_{\mathsf{f}\in\mathsf{F}}\{\mathbb{E}_T\left[h_\Omega(X)\cdot\mathsf{f}(\varphi(X))\right] - \mathbb{E}_S[Y\cdot\mathsf{f}(\varphi(X))]\}$ is the transferability error, F is the set of continuous functions from $\mathcal{Z}$ to $[-1,1]^C$, $\eta := \inf_{\mathsf{f}\in\mathsf{F}}\varepsilon_T(\mathsf{f}\varphi)$. Thus, to guarantee a small target error, the following conditions have to be met: a small source error of h_S (small $\varepsilon_S(h_S)$), a small transferability error of

h_Ω (small τ) and a strong inductive bias (small β) while we assume η small [7]. ADA incorporates a small set of target labelled samples into Ω to strengthen the inductive bias while enforcing a small transferability error of h_Ω. More details on the choice of Ω are given in Section 3.5.

4.2 A particular case with closed form

Setup and additional notations. In this section, we provide a simple example where the bound presented in Section 4 has a closed form. To conduct the analysis, we consider $\mathcal{X}$ as a measurable set provided with a probability measure noted p_T. We present an extension of an annotation selection to a measurable set. Selecting samples for annotation with budget b consists in determining some measurable subset $\mathcal{B}$ such that $p_T(X \in \mathcal{B}) = b$. In the particular case where $p_T := \sum_{x \in \mathcal{D}_T} \delta_x$ (δ_x is the Dirac distribution in x) is an empirical distribution, determining some measurable subset $\mathcal{B}$ such that $p_T(X \in \mathcal{B}) = b$ consists in determining a subset of b samples of $\mathcal{D}_T$.

Naive Active Classifier. Given a classifier h and an annotated subset $\mathcal{B}$ (with probability b), we suggest a slight modification of the classifier h based on the annotation provided by the Oracle of $\mathcal{B}$. To this purpose, we introduce the *naive active classifier*, noted $h_\mathcal{B}(x)$, and defined as follows:

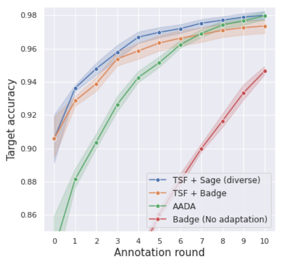
$$h_\mathcal{B}(x) = \text{Oracle}(x) \text{ if } x \in \mathcal{B}, h(x) \text{ otherwise.} \tag{12}$$

Thus, $h_\mathcal{B}(x)$ returns the classifier's output $h(x)$ if x is not annotated and returns the oracle's output Oracle(x) if x is annotated.

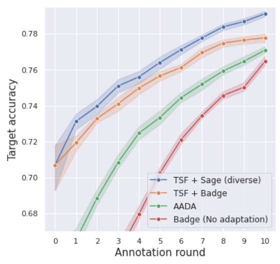
A closed bound. We want to exhibit a closed form of ρ when considering the active classifier. To this purpose, we introduce the *purity* π of $\mathcal{B}$, $\pi := p_T(h_S(X) \neq \text{Oracle}(X) | X \in \mathcal{B})$. It reflects our capacity to identify misclassified target samples. With this notion, we observe that the naive classifier improves the target error; $\varepsilon_T(h_\mathcal{B}) \leq \varepsilon_T(h_S) - b\pi$. Put simply, the error is reduced by $b\pi$ corresponding to annotated samples for which the prediction is different from the Oracle output. The higher the budget of annotation b and the higher the purity π, the lower the target error of the naive classifier. It corresponds to $\varepsilon_T(h_S) - b\pi = \left(1 - \frac{b\pi}{\varepsilon_S(h_S)}\right)\varepsilon_S(h_S) \leq (1 - b\pi)\varepsilon_S(h_S)$; resulting into $\beta = (1 - b\pi)$, and finally:

$$\varepsilon_T(h_\mathcal{B}) \leq \left(\frac{1}{b\pi} - 1\right)(\varepsilon_S(h_S) + 8\tau + \eta) \tag{13}$$

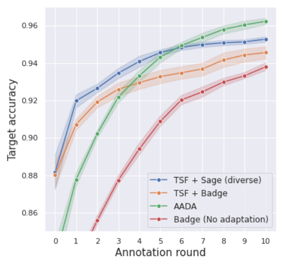
The target error of the active classifier is a decreasing function of both the purity and the annotation budget and an increasing function of the transferability error. The budget b, the purity π and the transferability of representations τ are levers to improve the naive classifier target error. The budget b must be considered as a cost constraint and not as a parameter to be optimized. The purity of π is not tractable since it involves labels in the target domain. Some proxy measures,



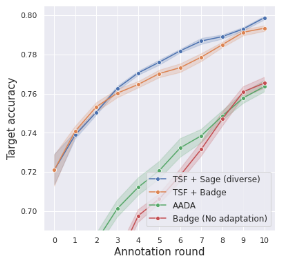
(a) A→W



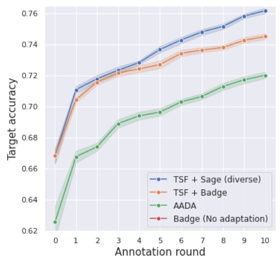
(b) W→A



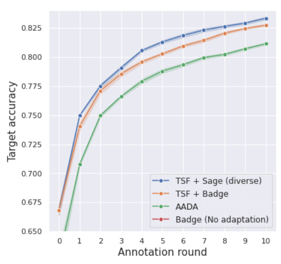
(c) A→D



(d) D→A



(e) VisDA($b = 16$)



(f) VisDA($b = 128$)

Fig. 3. Annotation of target samples improves adaptation drastically for the considered tasks. TSF+Sage (in blue) improves upon the state-of-the-art of ADA (AADA, in green), except for task A→D. AL (Badge, in red) performs poorly in this context (Badge without adaptation does not appear on VisDA tasks since it performs poorly: 47.0% and 63.4% after 10 rounds of annotation for $b = 16$ and $b = 128$, respectively) showing the importance of addressing the problem of adaptation for AL under distribution shift. Naively combining Badge with TSF (TSF+Badge, in orange) performs worsen than Sage. Sage takes into account the problem of domain shift when querying samples.

such as the entropy of predictions [14], can provide a fair estimation of purity. However, it is known that deep nets tend to be overconfident on misclassified samples [12]. Therefore, we focus our efforts on understanding the role of active annotation in improving transferability error τ.

5 Experiments

5.1 Setup

Tasks. We evaluate our approach on **Office-31** [29], **VisDA-2017** [25] and **DomainNet** [24]. Office-31 contains 4,652 images classified in 31 categories across three domains: Amazon (**A**), Webcam (**W**), and DSLR (**D**). We explore tasks **A** → **W**, **W** → **A**, **A** → **D** and **D** → **A**. We do not report results for tasks **D** → **W** and **W** → **D** since these tasks have already nearly perfect results in UDA [22]. For VisDA, we explore **Synthetic**: 3D models with different lighting conditions and different angles; **Real**: real-world images. We explore the **Synthetic** $\rightarrow$ **Real** task. **DomainNet** [24] is a large scale dataset with six domains and 345 classes (Clipart (**C**), Infograph (**I**), Painting (**P**), Quickdraw (**Q**), Real (**R**) and Sketch (**S**)). As **DomainNet** suffers of noisy labels, thus violates the assumption of a perfect Oracle, we focus on the subset of 126 classes and the 7 tasks **R**$\rightarrow$**C**, **R**$\rightarrow$**P**, **P**$\rightarrow$**C**, **C**$\rightarrow$**S**, **S**$\rightarrow$**P**, **R**$\rightarrow$**S** and **P**$\rightarrow$**R** [31].

Protocol. The standard protocol in UDA uses the same target samples during train and test phases. In AL's context, this induces an undesirable effect where sample annotation mechanically increases the accuracy. At train time, the model has access to input and label of annotated samples which are also present at test time. We suggest instead to split the target domain into a *train target domain* (samples used for adaptation and pool of data used for annotation) and *test target domain* (samples used for evaluating the model) with a ratio of 1/2. Therefore, samples from the test target domain have never been seen at train time. As a result, our protocol evaluates the model generalization in an inductive scenario. Reported results are based on 8 seeds for each method.

Budget, rounds and backbone. As the selected datasets are of different volumetry and difficulty, we used different budgets b: $b = 8$ for A$\rightarrow$W and A$\rightarrow$D (referred to as *easy tasks*), $b = 16$ for W$\rightarrow$A and D$\rightarrow$A (referred to as *medium tasks*), both $b = 16$ and $b = 128$ for VisDA (referred to as *hard tasks*). This allows to appreciate versatility of methods in *small* ($b = 8$), *medium* ($b = 16$) and *high* ($b = 128$) budget regimes. We conduct 10 rounds of annotation for these tasks. Additional details for DomainNet experiments are provided in comparison with SSDA in Section 5.2. Our backbone is a ResNet50 [16] trained by 10k steps of SGD by UDA before annotation. We use DANN [13] for AADA, MME [31] for MME based methods and TSF [7] for TSF based methods.

Baselines. **AADA** [34] is the closest algorithm to Sage. AADA adapts representations by fooling a domain discriminator d trained to output 1 for source data and 0 for target data [13] and scores target samples x; $s(x) := H(\hat{y})w(z)$ where $H(\hat{y})$ is the entropy of predictions $\hat{y}$ and $w(z) = (1 - d(z))/d(z)$. $H(\hat{y})$ brings information about uncertainty while $w(z)$ brings diversity to the score. We have reproduced the implementation of AADA. To demonstrate the effectiveness of Sage for Active DA, we report TSF with **Badge** query [3] (**TSF+Badge**), which is the state-of-the-art query in AL. For these methods, we used $\Omega = \Omega_{S+T}$. To compare Sage with an AL method which ignores domain shift between labelled samples and queried samples, we report **Badge** with $\Omega_{S\cup T}$. Finally, to compare with SSDA approaches, we build two methods upon **MME** [31] with **Entropy** query (selection samples with highest prediction entropy [35]), noted **MME+Entropy**, which is the most natural query for MME since it relies on max/min entropy, and with **Random** query noted **MME+Random**. We have reproduced the implementation of MME.

5.2 Results

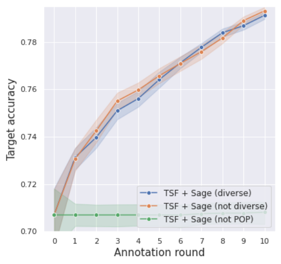
Comparison with SOTA. Results are reported in Figure 3. First, active annotation brings substantial improvements to UDA (round 0 of annotation). This validates the effort and the focus that should be put on ADA, in our opinion. Sage outperforms the current state-of-the-art (AADA) with a comfortable margin for tasks with medium or hard difficulty, except for tasks A$\rightarrow$D after the 5-th round. Importantly, Sage performs similarly or better than naively combining TSF with a state-of-the-art query in AL (Badge) demonstrating that Sage takes into account the problem of domain shift in the query process. Finally, using a direct AL method (Badge) fails in the context of domain shift.

Ablation of Sage. We ablate the core components of Sage *i.e.*, POP and the `k-means++` in Figures 4(a) and 4(b). Interestingly, Sage without POP fails to improve performances in the target domain. This demonstrates that POP brings information about uncertainty into the embedding. Sage without diversity performs poorly on VisDA($b = 128$), demonstrating that `k-means++` based sampling brings diversity. Diversity on Sage has a small effect on W$\rightarrow$A.

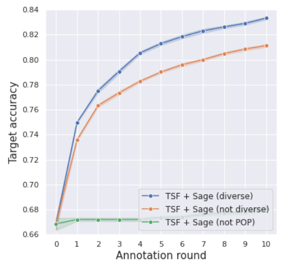
Ablation of queries. We ablate in Figures 4(c) and 4(d) more AL strategies : (**Random**), where target samples are selected at random, **Clusters** that selects the closest samples to b clusters of representations obtained with k-means, **Entropy** based on the highest entropy $\max_{x \in \mathcal{U}_T} -h(x) \cdot \log h(x)$ [35] and **Confidence** that used the smallest confidence $\min_{x \in \mathcal{U}_T} \max_c h(x)_c$ [35]. Sage is compared with a wide spectrum of AL queries based on representative (Random), diversity (Clusters) and uncertainty sampling (Entropy, Confidence). Sage outperforms them substantially on the two tasks, demonstrating it is well-suited for ADA.

Ablation of Ω. We report **TSF+$\boldsymbol{\Omega}_{\mathbf{S \cup T}}$** and **TSF+$\boldsymbol{\Omega}_{\mathrm{MME}}$** which consists in adding MME as a regularization of TSF *i.e.*, Ω used here is $\Omega_{S \cup T}$ and Ω_{MME}, respectively. Results are reported in Figures 4(e) and 4(f). We observe that using Ω_{S+T} and Ω_{MME} improve consistently wrt $\Omega_{S \cup T}$ on VisDA($b = 128$) while performing similarly on W$\rightarrow$A. Furthermore, we observe that adding MME to TSF+Sage achieves the best performances on VisDA($b = 128$). Importantly, MME+Entropy is already strong for VisDA($b = 128$) explaining the substantial improvement when adding MME to TSF for this task.

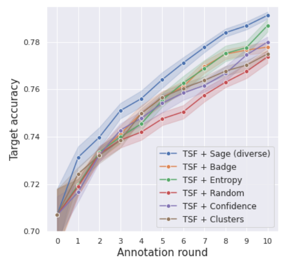
ADA vs SSDA: ADA is a more realistic setting. We compare SSDA (a fixed number of labelled target samples per class are available) with ADA (an Oracle provides ground-truth for queried target samples) when the number of target labelled samples are equal. Crucially, enforcing a fix number of labelled samples per class is unrealistic in practice. We report performances on DomainNet of MME (1 and 3 shot) [31] and Sage (here we used TSF + Sage + Ω_{MME}). AL is performed during 6 rounds with $b = 21$ and $b = 63$ for 1 and 3 shot respectively,



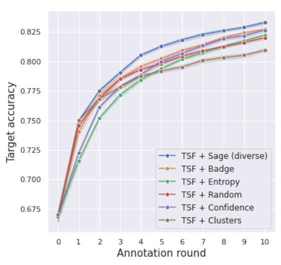
(a) W→A



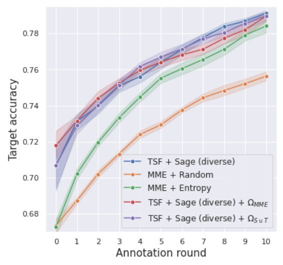
(b) VisDA($b = 128$)



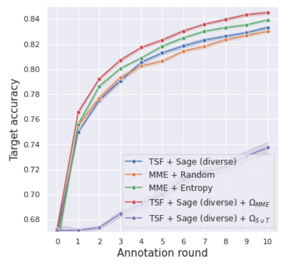
(c) W→A



(d) VisDA($b = 128$)



(e) W→A



(f) VisDA($b = 128$)

Fig. 4. (a) and (b): Both the POP and `k-means++` are crucial components for the empirical success of Sage. (c) and (d): Sage outperforms AL query based on representative, diversity and uncertainty samplings. (e) and (f): Effect of adding MME to TSF+Sage.

leading to the same number of target labelled samples[5]. Results are presented in Table 1. In the 3-shot scenario Sage improves upon MME on all the tasks, except P→R. In the 1-shot scenario, Sage and MME perform similarly. This demonstrates that active annotation with Sage performs equally, or better, than MME, and benefits from more realistic assumptions.

6 Related works

Transferability of Invariant Representations. Recent works warn that domain invariance may deteriorate transferability of invariant representations [18,38]. Prior works enhance their transferability with multi-linear conditioning of representations with predictions [22], by introducing weights [8,37,11], by penalizing high singular value of representations batch [10] or by hallucinating consistent target samples for bridging the domain gap [20].

Active Learning. There is an extensive literature on Active Learning [33] that can be divided into two schools; *uncertainty* and *diversity*. The first aims to annotate samples for which the model has uncertain prediction *e.g.*, samples are selected

[5] $|\mathcal{L}_T| = 21 \times 6 = 126$ (1 shot) and $|\mathcal{L}_T| = 63 \times 6 = 3 \times 126$ (3 shot)

Tasks	**1-shot**			**3-shot**		
	MME	AADA	Sage	MME	AADA	Sage
R→C	67.5	64.4	**69.3**	70.1	68.8	**73.9**
R→P	**69.6**	65.5	69.4	70.8	67.0	**71.4**
P→C	69.0	63.2	**69.9**	71.4	67.3	**74.1**
C→S	**62.2**	57.4	61.5	64.7	60.1	**65.4**
S→P	**67.9**	62.6	**67.9**	69.6	64.9	**69.8**
R→S	61.2	57.0	**62.1**	63.6	59.9	**65.8**
P→R	**79.3**	74.9	79.0	80.9	76.9	**81.2**
Mean	68.1	63.6	**68.5**	70.2	66.3	**71.7**

Table 1. SSDA (MME) vs ADA (AADA and Sage) on DomainNet. MME's results deviate from [31] due to train/test split, ResNet50 as backbone and minor implementation changes.

according to their entropy [35] or prediction margin [28], with some theoretical guarantees [15,4]. The second focuses on annotating a representative sample of the data distribution *e.g.*, the Core-Set approach [32] selects samples that geometrically cover the distribution. Several approaches also propose a trade-off between uncertainty and diversity, *e.g.*, [17] that is formulated as a bandit problem. Recently, the work [2] introduces Badge, a gradient embedding, which, like SAGE, takes the best of uncertainty and diversity. Our work is inspired by Badge and adapts the core ideas in the context of ADA.

Active Domain Adaptation. Despite its great practical interest, only a few previous works address the problem of *Active Domain Adaptation*. [9] annotates target samples by importance sampling while [27,30] annotates samples with high discrepancy with source samples based on the prediction of a domain discriminator. However, those strategies do not fit modern adaptation with deep nets. To our knowledge, AADA [34] is the only prior work that learns actively domain invariant representations and achieves the state-of-the-art for Active Domain Adaptation. AADA is the most relevant work to compare with Sage.

7 Conclusion

We have introduced Sage, an efficient method for ADA which identifies target samples that are likely to improve representations' transferability when annotated. It relies on two core components; a stochastic embedding of the gradient of the transferability loss and a `k-means++` initialization, which guarantees that each annotation round annotates a diverse set of target samples. Through various experiments, we have demonstrated the effectiveness of Sage and its capacity to take the best of uncertainty, representative, and diversity sampling. New SSDA strategies when using Sage is an interesting direction for future works.

References

1. Arthur, D., Vassilvitskii, S.: k-means++: The advantages of careful seeding. Tech. rep., Stanford (2006)
2. Ash, J.T., Zhang, C., Krishnamurthy, A., Langford, J., Agarwal, A.: Deep batch active learning by diverse, uncertain gradient lower bounds. arXiv preprint arXiv:1906.03671 (2019)
3. Ash, J.T., Zhang, C., Krishnamurthy, A., Langford, J., Agarwal, A.: Deep batch active learning by diverse, uncertain gradient lower bounds. In: 8th International Conference on Learning Representations, ICLR 2020. OpenReview.net (2020)
4. Balcan, M.F., Beygelzimer, A., Langford, J.: Agnostic active learning. Journal of Computer and System Sciences **75**(1), 78–89 (2009)
5. Beery, S., Van Horn, G., Perona, P.: Recognition in terra incognita. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 456–473 (2018)
6. Ben-David, S., Blitzer, J., Crammer, K., Pereira, F.: Analysis of representations for domain adaptation. In: Advances in neural information processing systems. pp. 137–144 (2007)
7. Bouvier, V., Very, P., Chastagnol, C., Tami, M., Hudelot, C.: Robust domain adaptation: Representations, weights and inductive bias. ECML-PKDD (2020)
8. Cao, Z., Ma, L., Long, M., Wang, J.: Partial adversarial domain adaptation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 135–150 (2018)
9. Chattopadhyay, R., Fan, W., Davidson, I., Panchanathan, S., Ye, J.: Joint transfer and batch-mode active learning. In: International Conference on Machine Learning. pp. 253–261 (2013)
10. Chen, X., Wang, S., Long, M., Wang, J.: Transferability vs. discriminability: Batch spectral penalization for adversarial domain adaptation. In: International Conference on Machine Learning. pp. 1081–1090 (2019)
11. Tachet des Combes, R., Zhao, H., Wang, Y.X., Gordon, G.J.: Domain adaptation with conditional distribution matching and generalized label shift. Advances in Neural Information Processing Systems **33** (2020)
12. Corbière, C., THOME, N., Bar-Hen, A., Cord, M., Pérez, P.: Addressing failure prediction by learning model confidence. In: Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., Garnett, R. (eds.) Advances in Neural Information Processing Systems 32, pp. 2902–2913. Curran Associates, Inc. (2019)
13. Ganin, Y., Lempitsky, V.: Unsupervised domain adaptation by backpropagation. In: International Conference on Machine Learning. pp. 1180–1189 (2015)
14. Grandvalet, Y., Bengio, Y.: Semi-supervised learning by entropy minimization. In: Advances in neural information processing systems. pp. 529–536 (2005)
15. Hanneke, S., et al.: Theory of disagreement-based active learning. Foundations and Trends® in Machine Learning **7**(2-3), 131–309 (2014)
16. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 770–778 (2016)
17. Hsu, W.N., Lin, H.T.: Active learning by learning. In: Twenty-Ninth AAAI conference on artificial intelligence. Citeseer (2015)
18. Johansson, F., Sontag, D., Ranganath, R.: Support and invertibility in domain-invariant representations. In: The 22nd International Conference on Artificial Intelligence and Statistics. pp. 527–536 (2019)

19. Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. In: Advances in neural information processing systems. pp. 1097–1105 (2012)
20. Liu, H., Long, M., Wang, J., Jordan, M.: Transferable adversarial training: A general approach to adapting deep classifiers. In: International Conference on Machine Learning. pp. 4013–4022 (2019)
21. Long, M., Cao, Y., Wang, J., Jordan, M.I.: Learning transferable features with deep adaptation networks. In: Proceedings of the 32nd International Conference on International Conference on Machine Learning-Volume 37. pp. 97–105. JMLR. org (2015)
22. Long, M., Cao, Z., Wang, J., Jordan, M.I.: Conditional adversarial domain adaptation. In: Advances in Neural Information Processing Systems. pp. 1640–1650 (2018)
23. Pan, S.J., Yang, Q.: A survey on transfer learning. IEEE Transactions on knowledge and data engineering **22**(10), 1345–1359 (2009)
24. Peng, X., Bai, Q., Xia, X., Huang, Z., Saenko, K., Wang, B.: Moment matching for multi-source domain adaptation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 1406–1415 (2019)
25. Peng, X., Usman, B., Kaushik, N., Hoffman, J., Wang, D., Saenko, K.: Visda: The visual domain adaptation challenge. arXiv preprint arXiv:1710.06924 (2017)
26. Quionero-Candela, J., Sugiyama, M., Schwaighofer, A., Lawrence, N.D.: Dataset shift in machine learning. The MIT Press (2009)
27. Rai, P., Saha, A., Daumé III, H., Venkatasubramanian, S.: Domain adaptation meets active learning. In: Proceedings of the NAACL HLT 2010 Workshop on Active Learning for Natural Language Processing. pp. 27–32. Association for Computational Linguistics (2010)
28. Roth, D., Small, K.: Margin-based active learning for structured output spaces. In: European Conference on Machine Learning. pp. 413–424. Springer (2006)
29. Saenko, K., Kulis, B., Fritz, M., Darrell, T.: Adapting visual category models to new domains. In: European conference on computer vision. pp. 213–226. Springer (2010)
30. Saha, A., Rai, P., Daumé, H., Venkatasubramanian, S., DuVall, S.L.: Active supervised domain adaptation. In: Joint European Conference on Machine Learning and Knowledge Discovery in Databases. pp. 97–112. Springer (2011)
31. Saito, K., Kim, D., Sclaroff, S., Darrell, T., Saenko, K.: Semi-supervised domain adaptation via minimax entropy. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 8050–8058 (2019)
32. Sener, O., Savarese, S.: Active learning for convolutional neural networks: A core-set approach. In: ICLR (2018)
33. Settles, B.: Active learning literature survey. Tech. rep., University of Wisconsin-Madison Department of Computer Sciences (2009)
34. Su, J.C., Tsai, Y.H., Sohn, K., Liu, B., Maji, S., Chandraker, M.: Active adversarial domain adaptation. In: The IEEE Winter Conference on Applications of Computer Vision. pp. 739–748 (2020)
35. Wang, D., Shang, Y.: A new active labeling method for deep learning. In: 2014 International joint conference on neural networks (IJCNN). pp. 112–119. IEEE (2014)
36. Yosinski, J., Clune, J., Bengio, Y., Lipson, H.: How transferable are features in deep neural networks? In: Advances in neural information processing systems. pp. 3320–3328 (2014)

37. You, K., Long, M., Cao, Z., Wang, J., Jordan, M.I.: Universal domain adaptation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 2720–2729 (2019)
38. Zhao, H., Des Combes, R.T., Zhang, K., Gordon, G.: On learning invariant representations for domain adaptation. In: International Conference on Machine Learning. pp. 7523–7532 (2019)